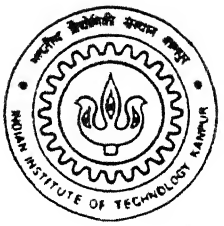
Path and Gait Generation of Legged Robots Using GA-Fuzzy Approach

by

Dilip Kumar Pratihar



DEPARTMENT OF MECHANICAL ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY KANPUR

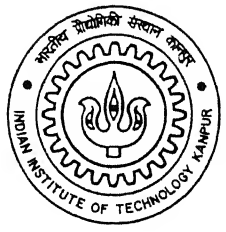
May, 1999

Path and Gait Generation of Legged Robots Using GA-Fuzzy Approach

A Thesis Submitted
in Partial Fulfilment of the Requirements
for the Degree of
Doctor of Philosophy

by

Dilip Kumar Pratihar



to the

DEPARTMENT OF MECHANICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY KANPUR

May, 1999

Certificate

It is certified that the work contained in the thesis entitled "**Path and Gait Generation of Legged Robots Using GA-Fuzzy Approach**" by Dilip Kumar Pratihar, has been carried out under our supervision and that this work has not been submitted elsewhere for a degree.

Kalyanmoy Deb 26/4/99

Dr. Kalyanmoy Deb
Associate Professor
Mechanical Engineering Dept.,
Indian Institute of Technology,
Kanpur 208 016.
India.

A Ghosh 11.4.99

Dr. Amitabha Ghosh
Professor
Mechanical Engineering Dept.,
Indian Institute of Technology,
Kanpur 208 016.
India.

Date:

Synopsis

Name of the Student :	Dilip Kumar Pratihar; Roll No.: 9610571
Degree for which submitted :	Ph.D.; Department: Mechanical Engineering.
Thesis Title :	"Path and Gait Generation of Legged Robots Using GA-Fuzzy Approach"
Name of Thesis Supervisors :	Prof. Amitabha Ghosh Prof. Kalyanmoy Deb
Month & Year of Submission :	May, 1999

In the present work, path generation and gait generation problems of a legged robot have been solved by using the combined genetic algorithm (GA) and fuzzy logic techniques (GA-Fuzzy approaches). A versatile and computationally tractable (suitable for on-line implementation) algorithm has also been developed based on the same GA-Fuzzy combination to tackle the problem of combined path and gait generation simultaneously of a six-legged robot. A GA is a population-based search and optimization technique which works based on the mechanics of natural genetics. On the other hand, a fuzzy logic controller (FLC) works based on fuzzy set theory and it is a powerful tool for dealing with uncertainty and imprecision. The working principles of both GA and FLC have been discussed in Chapter 2 of the thesis. In the proposed GA-Fuzzy approaches, these two potential tools have been merged to get advantages from both of them and to eliminate their individual limitations. Research is going on in both the directions - in one approach, an FLC is used to improve the performance of a GA, whereas in the other implementation, a GA has been used to improve the performance of an FLC. In this work,

both approaches have been studied in detail.

A considerable amount of work had been carried out, in the past, to solve the path generation problems (motion planning problems) of robot. Both graphical as well as analytical approaches had been tried by many investigators to solve these problems. Chapter 1 of the thesis provides an extensive survey on these techniques. The main drawback of these methods is their high computational complexity. Moreover, each of these methods is suitable for solving a particular type of problems. As an optimization module is not used in most of the techniques, the generated collision-free path may be far from being an optimal. Thus, there is still a need for the development of a versatile and computationally tractable algorithm to solve the time-minimal path planning problems of a mobile robot.

Chapter 3 describes a *fuzzy-genetic algorithm* in which a fuzzy logic technique has been used to create an initial population for GA and to modify the different GA parameters, namely crossover and mutation. The effectiveness of the proposed algorithm has been studied for solving the *find-path* problems of a mobile robot in which a robot will have to find a time-minimal collision-free path while moving among stationary obstacles. The same find-path problems have also been solved by using two other techniques, namely a *steepest descent method with a penalty function approach* and a *tangent graph technique along with A* algorithm.* The proposed algorithm is found to perform better than the steepest descent method with a penalty function approach. Moreover, the solutions of the proposed algorithm are similar to those of the best-known tangent graph and A* algorithm. It is to be noted that the performance of the proposed algorithm can be further improved by proper tuning of knowledge base of the fuzzy logic controller. It is also important to note that the proposed algorithm has linear computational complexity with respect to the number of intermediate control points, whereas the combined tangent graph and A* algorithm has quadratic computational complexity. Thus, the fuzzy-genetic algorithm is computationally faster than the combined tangent graph and A* algorithm.

A *genetic-fuzzy system* has been developed in which the performance of an FLC is improved by using a GA-based tuning. The proposed genetic-fuzzy system has been used to solve the *navigation problem* of a mobile robot in which a robot will have to determine its collision-free, time-optimal path while moving in the presence of moving obstacles (Chapter 4 of the thesis). There are six different approaches studied here, as described below:

1. Approach 1: Author-defined fuzzy logic controller.
2. Approach 2: Tuning scaling factors of the state variables alone.
3. Approach 3: Tuning rule base alone.
4. Approach 4: Tuning scaling factors and rule base in stages.
5. Approach 5: Tuning scaling factors and rule base simultaneously.
6. Approach 6: Automatic two-stage design of fuzzy rules using GA.

In the proposed genetic-fuzzy approach, the tuning of knowledge base of an FLC is done off-line, by using GA as an optimizer and once the optimal knowledge base is obtained, a mobile robot can use it on-line, to navigate in real-world similar scenarios, in an optimal sense. The performances of all the above mentioned six approaches have been compared. The GA-tuned FLC is found to perform better than the manually constructed FLC. The optimized solutions are obtained during the optimization of rule base only and tuning of scaling factors (which indicates the base width of triangular membership functions) cannot bring a significant improvement in the solution. Thus, optimizing rule base of an FLC is a rough tuning process, whereas optimizing scaling factors of the state variables is a fine-tuning process. The performance of an FLC depends on its knowledge base and designing a good knowledge base is not an easy task. To overcome this, a technique (Approach 6) has been developed in which a GA will design a good rule base of an FLC and no time is spent on manual construction of the fuzzy rule base. The results obtained using this approach are also found to be similar to those of the Approaches 3-5. However, Approach 6 is a more flexible approach in which the rule base of an FLC is designed automatically by using a GA.

The gait generation problem of a legged robot had been studied by many researchers. A detailed survey has been made in Chapter 1 of the thesis. The traditional methods (which include both graphical as well as analytical) of gait planning are computationally expensive. Moreover, the generated gait may not be optimal in any sense. In gait generation problem, some of the variables may be discrete in nature and handling of these variables becomes difficult in the traditional methods. Moreover, some attempts were made by the investigators to solve the problem of combined path and gait generation but their methods were able to provide only a partial solution to this complicated task.

Thus, an efficient and computationally tractable (suitable for on-line implementation) algorithms are yet to be developed which can tackle the path planning and gait planning problems of a legged robot simultaneously, in an optimal sense.

The periodic gait generation problem of a six-legged robot has been studied in Chapter 5 of the thesis. As the periodic (wave) gait is optimal in terms of static stability and the number of ground-legs is fixed, no optimization is carried out to further reduce the number of ground-legs. The problems of optimal crab gait generation while crossing a ditch and turning gait generation have been studied in Chapters 6 and 7, respectively, by using the similar genetic-fuzzy system. In this study, a six-legged robot will have to plan its gait in an optimal sense (with minimum number of ground-legs having the maximum average kinematic margin). Here, only two approaches have been considered, namely (i) Approach 1: Author-defined fuzzy-logic controller; (ii) Approach 2: Optimizing rule base alone. The performances of these two approaches are compared and Approach 2 is found to perform better than the Approach 1. Thus, GA-tuned FLCs have performed better than the author-defined FLCs.

In practice, a six-legged robot will have to find a collision-free, time-minimal path and to plan its stable gait simultaneously, in optimal sense (with minimum number of ground-legs having the maximum average kinematic margin) while moving on flat terrain. Three steps are to be considered for legged-robot locomotion - *determination of vehicle's trajectory*, *foothold selection*, and *design of a sequence of leg movements*. Thus, path and gait generations of a legged vehicle are to be done simultaneously. It is a complicated task and no single traditional approach is found to be successful in handling the problem. In the proposed algorithm (genetic-fuzzy system), a genetic algorithm (GA) is used to find optimized FLCs and optimal path and gaits are generated by using these optimized FLCs (Chapter 8 of the thesis). The performance of an FLC depends mainly on its rule base selection and optimizing membership function distribution is a fine tuning process. The hexapod will have to move along a straight path (periodic gait), to take a circular turn (non-periodic gait), to cross a ditch (non-periodic gait) as the situation demands. There is one FLC for determining the collision-free, time-optimal path and each leg of the hexapod is controlled by a separate FLC. Thus, there are seven FLCs running in parallel. Two different approaches are studied here. In one approach (Approach 1), the author-defined knowledge base of the FLC has been considered, whereas in the other approach (Approach 2), the rule base of the FLC has been optimized keeping its membership

function distribution unaltered. A binary-coded GA (1 for presence and 0 for absence of a fuzzy rule) is used to represent a solution. In all cases, the GA-tuned FLCs are found to perform better than the author-defined FLCs. It happens because the author-defined knowledge base of an FLC may not be optimal in any sense. The proposed algorithm is able to solve the problem of combined path and gait generation of a hexapod effectively. As optimization (tuning) is done off-line, the proposed algorithm is suitable for solving the similar problems on-line, in an optimal sense. As an FLC is less expensive computationally, the proposed algorithm will be computationally quicker (execution time is found to be only 1.0 sec in a HP 9000/K200 machine) compared to the traditional methods of combined path and gait generation. Finally, conclusions of the current study have been drawn and the scope for future work has been suggested in Chapter 9 of the thesis.

Dedicated to

my wife, Arpita
and son, Mimo

Acknowledgements

It is my proud privilege to express my deep sense of gratitude towards my supervisors, Prof. Amitabha Ghosh and Prof. Kalyanmoy Deb, for suggesting this work as well as for their invaluable guidance and constant encouragement throughout the course of the work. My introduction to the fields - *Robotics* and *Soft Computing* was entirely due to my advisors. Their inspiring guidance, systematic approach, stimulating discussions, sensible criticisms and extensive care helped to shape my thesis. I am profoundly grateful to both of them for their help and advice.

With gratitude and respect, I thank Prof. K. Muralidhar for his help and valuable suggestions. He was kind enough to act as my thesis caretaker.

I am grateful to Prof. H. Hatwal, the Head of the Centre for Robotics, for allowing me to carry out literature survey, to use computing and other facilities of the Centre. Special thanks are due to the other members of the Centre, namely Dr. B. Dasgupta, Dr. A. Mukherjee, Mr. S. Sen, Mrs. A. Kulkarni, Guptaji, Shuklaji and Pannaji. Their affection will be long remembered. My thanks are also due to Prof. S. G. Dhande, the In-charge of the CAD Project, the staff members and the students of the CAD Project for their help and cooperation.

I extend my profound thanks to my friends at Kanpur Genetic Algorithm Laboratory (KanGAL), namely Samir, Rajeev (both junior and senior), Sudipta, Alok, Nikhilesh. It is very difficult to forget the help I got from Samir during the course of my Ph.D. work. He designed and developed a small computer network at KanGAL which helped me to do everything (including computation, documentation, printing etc.) in our KanGAL itself. How can I forget the days and nights I spent sitting in front of the computer terminals at our little KanGAL ?

I am thankful to Prof. A. K. Mallik, Prof. V. Eswaran, Prof. G. Biswas, Prof. B. Sahay, Prof. P. Kumar, Prof. N. N. Kishore (all from the Department of Mechanical Engineering), Prof. N. Chakrabarty (of the Department of Metallurgical Engineering), Dr. P. Chakrabarty (of the Department of Civil Engineering) for their help and advice at various stages of my Ph.D. work.

I am indebted to my employer, Regional Engineering College, Durgapur, for granting me a study-leave throughout the course of my work. I am grateful to my colleagues at Durgapur for their help and cooperation. The financial help of the Quality Improvement Programme (Q.I.P.), Ministry of Human Resource and Development, Govt. of India, is also gratefully acknowledged.

I extend my thanks to Mrs. Ghosh, Mintu, Mrs. Deb for their help and cooperation which made our stay at I.I.T., Kanpur, a wonderful and a memorable one. Special thanks are also due to our little friends - Ronny and Bonny.

I dedicate this thesis to my wife, Arpita and son, Mimo. Arpita gave me a constant encouragement, helped me in so many ways during the period of my thesis work and without her help it would not have been a possibility. Our son, Mimo is also neglected during this period. I could spend enough time with him and we used to miss each other.

I extend my deepest gratitude to my parents, parents-in-laws and other family members for their invaluable love, affection, encouragement and support.

Last but not the least, I would like to express my sincere gratitude to all of them who directly and indirectly helped me for the successful completion of my thesis work.

Dilip Kumar Pratihar
Department of Mechanical Engineering
Indian Institute of Technology, Kanpur
India

Contents

List of Figures

List of Tables

Nomenclature

a	Acceleration of the robot
C	Number of ground-legs
d	Decoded value of binary string
d_n^m	Perpendicular distance from the center of m-th obstacle-circle to the n-th straight-line segment
d_{rem}	Eucleadian distance between the halted position of the robot and the destination point
D	Distance traveled by the robot along the trajectory
${}_B^W E$	Transformation vector from $\{W\}$ to $\{B\}$
f	Fitness
$\bar{f}$	Average fitness
f_{best}	Best fitness
f_{worst}	Worst fitness
F_r	Repulsive force
F_t	Attractive force
G	Final position of the robot
H	Total number of training scenarios
k	Kinematic margin of a ground-leg
K	Average kinematic margin of the ground-legs
${}^B L_i$	Position of i-th leg with respect to the body coordinate frame $\{B\}$
${}^W L_i$	Position of i-th leg with respect to the world coordinate frame $\{W\}$
l	String length of binary coded GA
M	Total number of obstacles
N	Total number of control points
P_1, P_2	Penalty terms
$P_{predicted}$	Predicted position of obstacle
$P_{present}$	Present position of obstacle
$P_{previous}$	Previous position of obstacle
p_c	Probability of crossover
p_m	Probability of mutation

$p(t)$	Predefined path
Q	Total number of motion segments
q	Exponent (positive real number)
R	Radius of curvature
${}^{W}_{B}R$	Rotation matrix
r^m	Radius of m-th obstacle circle
S	Initial position of the robot
s	Stability margin
s_L	Longitudinal gait stability margin
SF_a	Scaling factors for angle
SF_d	Scaling factors for distance
t	Cycle time
tr	Tolerance
T	Traveling time
T_{max}	Maximum traveling time
u	Random number
U_f	Output of the fuzzy logic controller
v	Maximum velocity of the robot along the trajectory
w_1, w_2	Weighting factors
X	Universe of discourse or universal set
(X_n, Y_n)	Intermediate control points
(X_0, Y_0)	Coordinate of initial position of the robot
(X_{N+1}, Y_{N+1})	Coordinate of final position of the robot
(X_c^m, Y_c^m)	Center of m-th obstacle circle
x^L	Lower bound of the variable, x
x^U	Upper bound of the variable, x

Abbreviations

A	Ahead
AL	Ahead left
ANN	Artificial neural network
AR	Ahead right
AVA	Average variance of the alleles
B	Behind
BAM	Bidirectional Associative Memory
BL	Behind left
BR	Behind right
CV	Certainty value
Ch	Child
CG	Center of gravity
DB	Data base
$DPGA$	Dynamic parametric genetic algorithm
EA	Evolutionary algorithm
ELF	Evolutionary learning of fuzzy rules
FLC	Fuzzy logic controller
FL	Fuzzy logic
$FFNN$	Feed-forward neural network
FR	Far
FS	Fitness of the string
FGA	Fuzzy genetic algorithm
GA	Genetic algorithm
GDM	Genotypic diversity measure
GFS	Genetic fuzzy system
KB	Knowledge base
LT	Left
LR	Large
m	Metre
m/sec	Metre per second
m/sec^2	Metre per second^2
MD	Moderate
MM	Medium
NL	Negative large
NM	Negative medium
NN	Neural network
NR	Near
PDM	Phenotypic diversity measures

$PBGA$	Pseudo-bacterial genetic algorithm
PL	Positive large
PM	Positive medium
PPP	Path planning problem
Pr	Parent
RB	Rule base
RF	Resultant force
RT	Right
sec	Second
SBX	Simulated binary crossover
SL	Slightly large
SM	Small
VAC	Variance average of the chromosomes
VF	Very far
VFF	Virtual force field
VFH	Vector Field Histogram
VL	Very large
VN	Very near
VPP	Velocity planning problem
VS	Very small
Z	Zero

Greek symbols

α	Spread factor
β	Duty factor
$\bar{\delta}$	Perturbance factor
δ_{max}	Maximum perturbance
η_1, η_2	Weighting factors
γ	Phase difference
$\bar{\gamma}$	Cumulative probability
μ	Membership function value
ψ_p	Angle of p-th straight-line segment with the horizontal axis
θ_n	Included angle between two consecutive straight-line segments

Chapter 1

INTRODUCTION

In this chapter, a detailed survey has been made on the different conventional methods (both graphical as well as analytical) of path and gait generation of a robot and their limitations are pointed out. The objective and scope of the current study have also been discussed in this chapter.

1.1 Introduction to Mobile Robots

Robot, a versatile mechanical device, is generally available in the form of a manipulator arm, a mobile vehicle, a free-flying platform, a multi-joint multi-fingered hand or a combination of these. There are basically two types of mobile robots, namely wheeled robot and legged robot. A wheeled robot is suitable for a smooth terrain and is more energy efficient and easier to control. On the other hand, a legged robot can operate well in hostile environments and promises a high degree of terrain adaptivity and maneuverability. However, it has a complex structure and poor controllability. Current research in robotics aims to build autonomous and intelligent robotic systems to meet the increasing industrial demand for automatic manufacturing systems. An autonomous robot should be able to perform the following tasks:

- Obtaining information of the environment using visual or auditory sensors,
- Building world model,

- Decision-making,
- Motion planning,
- Learning from past experience to improve performance.

An autonomous vehicle has the following advantages:

- It is capable of handling minor uncertainties. Thus, it need not be re-programmed always, if there is a minor change in the environment.
- It reduces the down-time needed for re-programming.
- It can work in hazardous environments.

Thus, one of the many features an autonomous robotic device should have is its ability to plan motion. Motion planning enables a robot to move in its environment securely while executing a given task. Motion planning methods can be either local or global, exact or heuristic in nature. A mobile robot will have to plan its obstacle-free path in varying situations while moving from an initial position to a final position. Thus, a mobile robot should have an *obstacle avoidance* scheme during its *navigation.*

1.1.1 Obstacle Avoidance and Navigation

An autonomous mobile robot should be able to plan its motion in varying situations in such a way that it can reach its destination by avoiding collision with all the obstacles. Thus, obstacle avoidance is an important area of research and it has a number of applications in designing an autonomous vehicle. Initially, the obstacle avoidance scheme was studied only for a perfectly known environment called *structured environment.* Now-a-days, research is going on for designing suitable obstacle avoidance schemes even for the environments which are unknown beforehand and liable to change suddenly. These environments are known as *unstructured environments.* In an unstructured environment, a mobile robot will have to plan its motion based on the available sensory information.

Navigation is defined as "the science of getting ships, aircraft or spacecraft from place to place, especially the method of determining position, course and distance travelled"

(Merriam-Webster 1985). The term - *navigation* has been used here in connection with an autonomous mobile robot. In general, navigation is the problem of deciding on a path or route to be followed. An autonomous vehicle should be capable of navigating in an unknown and dynamic environment. It is important to note that in a navigation problem, the search space for a robot is the plane of motion, whereas in a *manipulation* problem, the workspace of the manipulator is considered as its search space.

1.1.2 Complex Optimization and Soft Computing

Real-world problems are very complex in nature and most of the traditional optimization techniques (Deb 1995) will fail to find a global optimum solution in those cases. Moreover, an exact mathematical modeling is not possible for most of the real-world optimization problems. The gradient-based techniques fail because there may be discontinuity in the expression of objective function itself, in most of the real-world optimization problems. Sometimes, due to the complexity of the problem, determining a gradient of the objective function mathematically becomes a very difficult task. Moreover, in such a complex optimization problem, the variables may be of mixed type - some of the variables are discrete in nature, whereas the others are continuous variables.

Soft computing techniques, on the other hand, can solve the complex real-world problems within a reasonable accuracy. The computational complexity is also expected to be low due to their heuristic nature. Soft computing techniques include fuzzy logic (FL) technique, neural network (NN), genetic algorithm (GA) and others. A fuzzy logic controller is generally used for handling imprecision and uncertainty. Neural network is a potential tool for learning and genetic algorithm is a powerful tool for optimization. Sometimes, some of these techniques are combined to get advantages from each of them which is necessary to solve a complex practical problem. Thus, the combined techniques like GA-FL, GA-NN, FL-NN, GA-FL-NN have also been developed by several researchers to get a reasonable solution of a complex problem at a low computational cost.

1.2 Navigation of Mobile Robots

A mobile robot will have to plan its time-optimal path (in the plane of motion) from an initial position to a final position by avoiding a set of obstacles (either stationary or moving obstacles). Motion planning problems can be classified into two groups, namely motion planning with complete information and motion planning with incomplete information.

1.2.1 Motion Planning with Complete Information

It happens when the full information about the geometry of the robot and the stationary obstacles in the environment is given beforehand, so path planning becomes a one-time off-line operation. It is also known as a *global approach.* The following steps are followed in this approach:

- collect information about the task and the environment;
- find the complete solution to the task;
- begin task execution.

Here, the task execution is a preprogrammed rigid process with little information processing. It is also known as *Act-After-Thinking* process.

1.2.2 Motion Planning with Incomplete Information

A mobile robot will have to plan its path in the presence of some moving obstacles. The information about the environment is collected with the help of sensors. No detailed model of the environment is assumed and planning is done continuously (on-line) based on the available sensory information. It is also known as a *local approach* or *Act-While-Thinking* process. As sensor readings are not always precise, the planner should be able to deal with uncertainty and imprecision. In an unknown and dynamic environment, the following factors are to be considered during motion planning:

- sensing device;

- method of predicting obstacle motions;
- modeling method of obstacle motion;
- algorithms for real-time collision-free trajectory planning.

1.3 Motion of Multi-legged Robots

Legged locomotion has a longer history compared to that of wheeled locomotion. A legged vehicle is preferred to a wheeled vehicle particularly in case of uneven terrain although the control mechanism of the former is more complex compared to that of the latter. Legged vehicles have the following advantages over the wheeled vehicles as given below (Song and Waldron 1989):

- Higher speed;
- Better fuel economy;
- Greater mobility;
- Better isolation from terrain irregularities;
- Lesser environmental damage as a legged vehicle uses isolated footholds, whereas a wheeled vehicle requires a continuous path of support.

A statically stable multi-legged robot is generally used in the form of either a four-legged robot (quadruped) or a six-legged robot (hexapod). The six-legged robot has the following advantages over the four-legged robot:

1. A four-legged robot is more susceptible to deadlock situations (in which an unstable legged robot is unable to regain its stability by placing an extra leg on the ground) compared to a six-legged robot.
2. In case of heavy walking environment, a hexapod is preferred to a four-legged robot.
3. A six-legged robot is more robust compared to a four-legged robot because if one of the legs of a hexapod becomes out of order, it can still continue walking with the rest five legs.

It is important to note that a multi-legged robot will have to plan its path and gait (sequence of leg motions) simultaneously during its locomotion.

1.3.1 Path Generation Problem

A legged robot, during its locomotion, should not collide with any of the obstacles (either stationary or moving obstacles) present in its plane of motion. Thus, a multi-legged robot will have to plan its collision-free path along with its gait planning. An autonomous legged vehicle should have an obstacle avoidance scheme besides its gait planning scheme.

1.3.2 Gait Generation Problem

Besides path planning (for determining vehicle's trajectory), a multi-legged robot will have to determine its footholds and the possible sequence of leg movements. A *gait* is defined as a sequence of leg movements coordinated with a sequence of body motions for the purpose of transporting the body of the legged vehicle from an initial position to a final position. There are basically two types of gait pattern - *periodic* and *non-periodic* (Song and Waldron 1989). A periodic gait is suitable for a perfect (smooth) terrain. There are various types of periodic gait, namely *wave gait, equal phase gait*, and others (Song and Waldron 1989). In the wave gait, the placing and lifting of the legs follow a wave like pattern, whereas the placing and lifting events are distributed evenly over a locomotion cycle in an equal phase gait. Periodic gaits are optimal from the point of view of static stability. A periodic gait pattern is not suitable if the terrain is rough and a non-periodic gait pattern is used for this purpose. There are various types of non-periodic gait, namely *free gait, adaptive gait, discontinuous follow-the-leader gait, large-obstacle gait*, and others (Song and Waldron 1989). The free gait algorithm tries to maximize the number of legs in the air so that the legged vehicle can recover its stability by placement of one of the legs presently in the air. On the other hand, the adaptive gait always keeps the same number of legs on the ground, providing that the stability of the vehicle is maintained. Thus, the computation of the adaptive gait is faster than that of the free gait. In discontinuous follow-the-leader gait, the footholds selected by the front feet are followed subsequently by the middle and the hind feet. The advantage of the follow-the-leader gait is that the planner only needs to select the permitted footholds for the two front legs and the rest of

the legs automatically step on the permitted footholds. It has a good terrain adaptability but the maneuvers such as sideways stepping, turning are not so easy. The gait pattern used by a walking machine to cross large obstacles is known as a large obstacle gait.

1.3.3 Combined Path and Gait Generation

In practice, a legged vehicle should be able to plan its path and gait simultaneously. For legged locomotion, a vehicle will have to follow all the three steps simultaneously as stated below:

1. Determination of vehicle's trajectory;
2. Selection of footholds;
3. Determination of the sequence of leg motions.

1.4 Literature Review

1.4.1 Robot Motion Planning Among Static Obstacles

Several methods had been tried, in the past, by various investigators to solve the motion planning problem of a robot in the presence of static obstacles. Latombe (1991) provides an extensive survey on the different traditional methods used in motion planning. These methods include both graphical as well as analytical approaches. Graphical methods include *road map, cell decomposition* and others. The concept of *configuration space* (Lozano Pérez 1983, 1987) had been widely used in solving the motion planning problem. Configuration space is the parameter space of positions. The obstacles in this space are represented by values of parameters those cause collisions. The *free space* is defined as the sub-set of configuration space that is free of collisions. The path-planning problem then becomes the problem of finding a path between the initial and final positions, within the free space.

Road map is one of the earliest path planning methods. This method consists of capturing the connectivity of the robot's free space in a network of one-dimensional curves. Road map uses *visibility graph* (Nilsson 1969, Lozano-Pérez and Wesley 1969, Lozano-Pérez 1987), *Voronoi diagram* (Leven and Sharir 1987, O'Dunlaing et al. 1986, 1987), *freeway net* (Brooks 1983). Visibility graph is the non-directed graph whose nodes are the initial and final configurations and all the C-obstacle vertices and it connects those vertices of obstacles which are visible from one another (refer to Fig. 1.1), whereas Voronoi diagram represents the locus of points those are equidistant from at least two of the obstacle boundaries (refer to Fig. 1.2). Freeway method consists of extracting

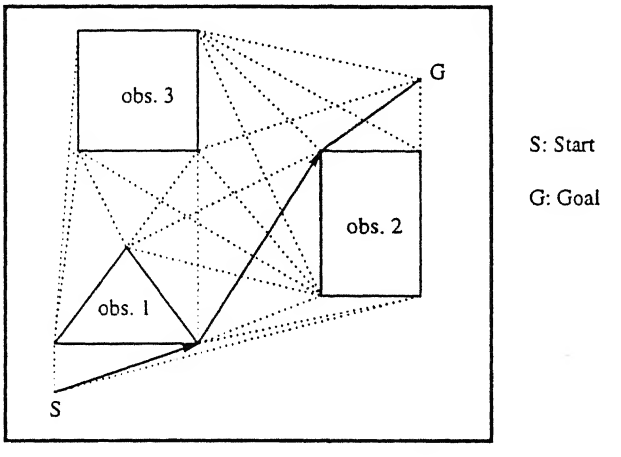


Figure 1.1: Path generated by visibility graph (shown by solid lines).

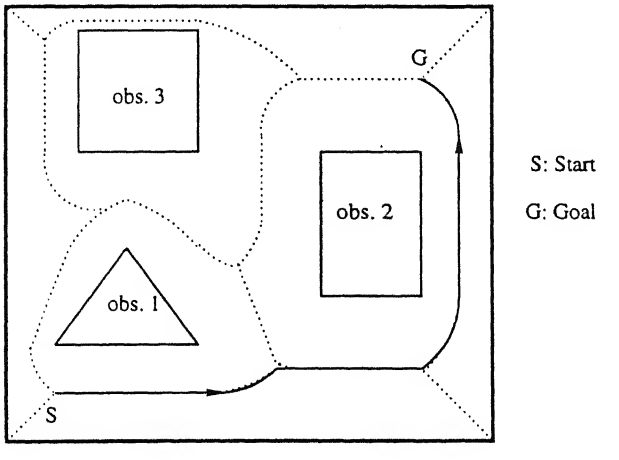


Figure 1.2: Path generated by Voronoi diagram (shown by solid lines).

geometric figures known as freeways (i.e. straight linear generalized cylinders) from the workspace, connecting them into a graph to form the freeway net and searching this graph.

In cell decomposition (Lozano Pérez 1983), the robot's free space is divided into a number of simple regions called cells. A connectivity graph between the cells is then constructed and searched. Fig. 1.3 shows a path generated by the cell decomposition

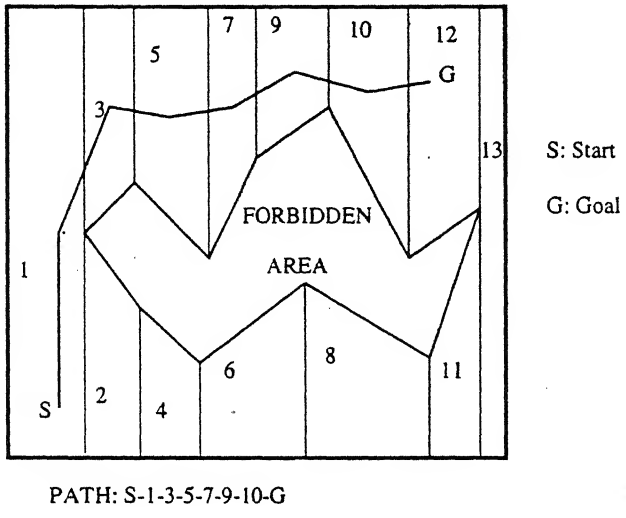


Figure 1.3: Path generated by cell decomposition method.

method. Moreover, a series of five papers called the Piano Movers' problem series were published by Schwartz et al. (1983a, 1983b, 1983c, 1983d) and Sharir and Ariel-Sheffi (1983). The main disadvantage of these methods is their computational complexity. Reif (1979) presented the first theoretical investigation of the inherent computational complexity of the motion planning problem, showing that planning a path among fixed obstacles is PSPACE - hard (i.e. the time required to solve the problem is exponential) in the dimension of configuration space.

Khatib (1986) pioneered the *potential field method* which is based on artificial potential produced by the goal configuration and the C-obstacles. Here, the robot is represented as a point in the configuration space and the goal configuration generates an attractive potential which pulls the robot towards the goal, whereas a repulsive potential is created by the C-obstacles which pushes the robot away from them. The robot moves under the combined action of these attractive and repulsive potentials. In this connection, work of Warren(1989), Kim and Khosla (1992), Rimon and Koditschek (1992), Koren and Borenstein (1991) are worth mentioning. Although the potential field approach is a widely used

method for solving this type of problems, it has the following disadvantages:

1. The solution largely depends on chosen potential function and it may get trapped into local minimum of the potential function. This happens when the attractive potential of the goal is balanced by the repulsive potential due to the presence of obstacles. Such problems are likely to occur also in environments that contain obstacles those which are non-convex in shape.

2. When the robot travels in a narrow corridor in which it experiences repulsive forces simultaneously from the opposite sides, the motion becomes unstable.

3. It is unable to find a path among closely spaced obstacles.

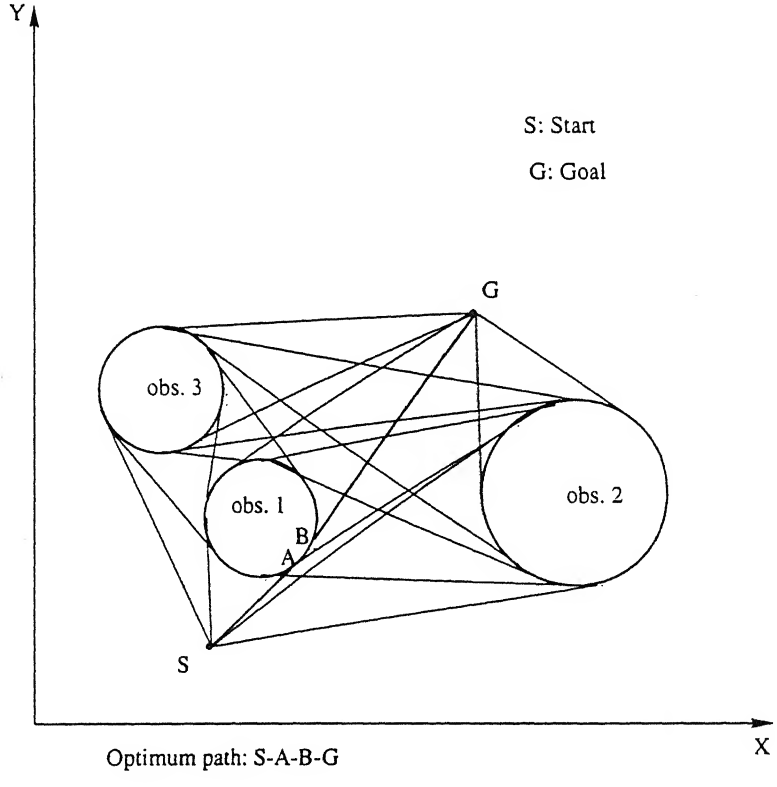


Figure 1.4: Optimum path generated by tangent graph along with A* algorithm.

Minimum-time path planning problems (in which the robot tries to find an obstacle-free path having minimum traveling time) had also been attempted by several investigators. In this connection, work of Kim and Shin (1984), Slotine and Yang (1989), Reister

and Lenhart (1995) are significant. Both the linear and nonlinear programming techniques had been used for optimization. Moreover, Liu and Arimoto (1991, 1992, 1995) proposed the *tangent graph method* along with A* algorithm to find time-optimal path in the presence of circular and non-circular obstacles. In the method of handling circular obstacles, tangents are drawn from the starting point to the nearest obstacle and from one obstacle to another. Thereafter, a search algorithm i.e. A*algorithm (Rich 1983) is used to find the path comprising of tangents and circular arcs (using obstacle boundaries) which make the total traveling time minimum (refer to Fig. 1.4). The complexity of their algorithm was found to be $O(N^2)$, where N is the number of convex segments of obstacle boundaries. Thus, the complexity of the combined tangent graph and A* algorithm is quadratic to the number of control points and the algorithm becomes computationally intractable and impractical to use for a reasonably large number of obstacles.

1.4.2 Robot Motion Planning Among Moving Obstacles

A considerable amount of work had been carried out, in the past, to develop suitable methods for robot motion planning in the presence of moving obstacles.

Kant and Zucker (1984, 1986, 1988) introduced a method named *path velocity decomposition* for solving the motion planning problems of a mobile robot in the presence of moving obstacles and later on, this approach was used by several other investigators (Fujimura and Samet 1989, Griswold and Eem 1990, Kyriakopoulos and Saridis 1991, Fraichard and Laugier 1993). In this approach, trajectory planning problem is decomposed into two subproblems as follows:

1. planning a path to avoid collision with static obstacles - it is known as path planning problem (PPP).

2. planning the velocity of the robot along the path to avoid collision with moving obstacles - it is known as velocity planning problem (VPP).

Thus, it is a two-stage method which provides an efficient solution to the problem. This heuristic decomposition results in a significant reduction in the complexity of the problem.

But, this method suffers from the following drawbacks:

1. This approach may fail to yield a collision-free trajectory even though it exists. For example, it will fail if an obstacle moves along the same path as the robot.
2. The path yielded by this method may not be good always, it may be unnecessarily a lengthy one if there are too many obstacles crossing the chosen path.
3. This approach will work best for obstacles ~~those~~ which are sparsely distributed and moving relatively orthogonally to the path of the robot.
4. As there is a sudden change of velocity, the robot will have a jerky motion which is not desirable.

In a dynamic environment, visibility graph will go on changing and a concept of *accessibility graph* had been introduced by Fujimura and Samet (1988) to solve the motion planning problems. The concept of accessibility is the generalization of the visibility concept. It is to be noted that this approach is successful in a limited domain.

Motion planning problems, in a dynamic environment, had also been solved by using the *space-time* concept (Reif and Sharir 1985, Kant and Zucker 1986, Erdmann and Lozano-Pérez 1986, Lamadrid and Gini 1990, Shin et al. 1990). Space-time is constructed by adding an extra dimension - *time* - to ~~it~~ it. In the space-time representation, the time-varying obstacle is converted into a static obstacle. Thus, the motion planning problems among moving obstacles are converted into path planning problems for stationary obstacle avoidance in the space-time representation. But, the complexity of the problem increases, as an extra dimension (time) is added to the space.

Incremental planning scheme was used by Slack and Miller (1987), Lamadrid (1994) to solve the motion planning problem among moving obstacles. This approach can be used to deal with obstacles that follow unpredictable trajectories. An incremental planning system would predict the motion of obstacles in the workspace and plan a path that would avoid them. This path would be followed until it became invalid (validity is checked based on sensory information). The system would then adjust its predictions and plan a new path. The new path would be executed again until it became invalid. This process would continue until the robot reached its destination.

A *probabilistic approach* had been proposed by Sharma (1992) to solve the motion planning problems in a partially known environment. This approach is able to deal with uncertainty. The dynamic component of the environment is represented as a set of discrete events which are detected on-line and the mobile robot is able to change its speed or path as the situation demands. To deal with the uncertainty of the environment, the events are modeled as stochastic processes following a *Poisson distribution*. Moreover, Zhu (1991) used a stochastic model (*Hidden-Markov process*) for forecasting of obstacle movements in a dynamic environment. More recently, Nam et al. (1996) have suggested a method in which the trajectories of the obstacles are predicted using a stochastic model of obstacle motion. The obstacle motion is modeled as a *random walk process*. The concepts of *view-time* and *view-period* have been introduced to discretize the motion of obstacles. View-time is defined as the time instant at which the robot senses the obstacle position and velocity. View-period is defined as the time interval during which the robot performs sensing, predicting and planning of collision-free motion. This method has the following characteristics:

1. It uses the concept of view-time to discretize the motion of obstacle;
2. It predicts the obstacle trajectory for a view-period using the random walk process;
3. It uses the artificial potential field method to plan a collision-free trajectory of the robot.

Fiorini and Shiller (1993) proposed one approach using the concept of *relative velocity* for solving the motion planning problems of mobile robot in a dynamic environment. Here, the original dynamic problem is converted into several static problems using the relative velocity between the robot and the obstacles. These static problems are converted into a single problem by means of a vector transformation. The set of velocity vectors are then computed so that the robot avoids collision with all the moving obstacles.

Fujimura (1995) used a two-level graph search technique to find out time-minimum routes in a time-dependent network. Due to the presence of dynamic objects in the environment, connectivity between the network changes over time, i.e. the network is time-dependent. The objective of this study is to generate a time-minimum route from an initial node to a destination node. It is to be noted that his algorithm has polynomial

time complexity with respect to the size of the network and the number of moving obstacles in the plane.

Lamadrid and Gini (1990) suggested the method of *path tracking* which is applicable in the presence of objects moving on unknown trajectories and with unknown velocities. Here, the robot must follow a predefined path $p(t)$, a continuous function of time, with a given tolerance and reach its destination within a given time without colliding with any obstacle. Thus, the robot is allowed to deviate from the desired path by a certain tolerance tr. This system requires a path as input which is produced by a path planning system considering the obstacles to be stationary.

The *potential field approach* had also been used by several researchers to solve the motion planning problems of a mobile robot among moving obstacles. In this connection, work of Suh and Shin (1988), Khosla and Volpe (1988), Okutomi and Mori (1986), Newman and Hogan (1987), Barraquand et al. (1992), Nam et al. (1995) are worth mentioning. In this approach, obstacles exert repulsive forces onto the robot, whereas the target applies an attractive force to the robot. The sum of all these forces i.e. the resultant force will determine the subsequent direction and speed of travel of the robot. Thus, the robot moves under the influence of this resultant force. This method is popular for its simplicity and elegance. But, it has a few disadvantages as discussed earlier (Section 1.4.1). To overcome some of these drawbacks of the artificial potential field method, two different approaches, namely *Virtual Force Field*, VFF (Borenstein and Koren 1989) and *Vector Field Histogram*, VFH (Borenstein and Koren 1990, 1991) had been proposed for real-time obstacle avoidance of mobile robots. These two approaches have been discussed here in detail.

1. **Virtual Force Field Method (VFF)**: It uses a two-dimensional Cartesian grid called the *histogram grid* for representing the obstacle. Each cell in the histogram grid holds a certainty value (CV) that represents the confidence of the algorithm in the existence of an obstacle at that location. In the histogram grid, CVs are incremented when the sensor reading indicates the presence of an obstacle at that cell. Thus, the active cells and active region are found out. Each active cell exerts a virtual repulsive force, F_r toward the robot. A virtual attractive force, F_t of constant magnitude is also applied to the robot, pulling it toward the target. Thus, the robot moves under the action of resultant force vector $RF = F_r + F_t$. It is to be

mentioned that this approach will fail when the clearance between two obstacles is less. Thus, a robot will not be able to pass through a door-way because the repulsive forces from both sides of the door-way may result in a force that will push the robot away. Moreover, the resulting motion becomes oscillatory and unstable when the robot travels in a narrow corridor.

2. **Vector Field Histogram (VFH)**: In VFF approach, a single-step drastic data reduction occurs when the individual repulsive forces from histogram grid cells are totaled to calculate the resultant force vector, F_r. Thus, the detailed information about the local obstacle distribution is lost. To overcome this shortcoming, in VFH approach, a two-stage data reduction is employed in place of single-stage data reduction of VFF approach. Here, the histogram grid is reduced to a 1-D *polar histogram* which helps in getting a spatial interpretation of the robot's instantaneous environment. It is important that only those sectors are selected for *candidate valleys* whose polar obstacle density is below the threshold value. The candidate valleys closest to the target is selected for further processing to determine the direction of movement of the robot. Moreover, the speed of the robot is reduced when it is approaching the obstacles head-on. It is to be noted that the VFH method does not attempt to find an optimal path. Moreover, a VFH-controlled robot may get trapped in dead-end situations.

Reactive control strategies were developed by several investigators to solve the motion planning problem of mobile robots (Brooks 1986, Arkin 1989, Gat 1991, Arkin and MacKenzie 1994). The reactive control scheme works based on the traditional artificial intelligence model of human cognition. This algorithm makes an elaborate world model based on the sensor reading and subsequently plan the robot's actions. It requires large amount of computation and decision making, resulting in a slow response. This approach is suitable for dynamic and unmodeled (or partially modeled) environments. There is a tight coupling between perception and motor action. Robotic actions are decomposed into a collection of primitive motor behaviors (Arkin 1990), each of which is capable of functioning more or less independently. These primitive behaviors include *move-to-goal*, *avoid-obstacle*, *avoid-past* and others. Multiple motor schemata (behaviors) act in a concurrent manner to yield a globally emergent behavior that strives to satisfy the robot's goal. But, this approach has the following disadvantages:

1. As the behaviors are hard-wired, this approach is unable to handle environments which the programmer did not foresee.

2. It requires large amount of computer memory.

In order to achieve more robust reactive control, *behavior switching* and *behavior adaptation* had also been adopted. Behavior switching is to dynamically select behaviors appropriate for a given environment and behavior adaptation is to adapt and fine-tune the existing behaviors dynamically. Moreover, attempts were made by many researchers (Mitchell 1990, Lin 1993, Mahadevan 1992, Chien et al. 1991) to improve the performance by integrating learning techniques with reactive control. The *case-based reactive navigation* schemes had also been proposed by many investigators (Kolodner 1990, Ram et al. 1997) in which a library of cases is maintained to provide suggestions for action in new situations.

A time-complexity study of the motion planning problem among moving obstacles was also presented by many researchers. Reif and Sharir (1985) showed that motion planning for a 3-D environment containing moving obstacles is PSPACE-hard given bounds on the robot's velocity and NP-hard without such bounds. Moreover, Canny and Reif (1987) proved that motion planning for a point in the plane with a bounded velocity is NP-hard, even when the moving obstacles are convex polygons moving at constant linear velocity without rotation.

1.4.3 Drawbacks of the Traditional Methods of Motion Planning

The traditional methods of robot motion planning have the following drawbacks:

1. The traditional methods are computationally expensive even for a simple problem. These methods become intractable computationally for a more complex motion planning problem.

2. Each of these conventional methods is suitable for a particular type of problems. Till now, there is no versatile algorithm which is applicable to all the problems.

3. As most of the algorithms do not have an optimization module, the generated path may not be optimal in any sense.

4. Although potential field method is the most widely used approach for solving the motion planning problem of robots, the robot may become trapped at local minima in the resultant potential field.

Thus, there is still a need for the development of an efficient, versatile and computationally tractable algorithm for solving the motion planning problems of robots.

1.4.4 Robot Motion Planning Using Fuzzy Logic Technique

To reduce the computational complexity of motion planning problem, heuristic-based methods were also developed by several investigators. In a dynamic environment, motion planning is based on sensor readings and future prediction of location of the obstacles. Sensor readings are associated with imprecision and uncertainty (Ferrari and Chemello 1990). Therefore, a *fuzzy logic controller* (refer to Section 2.2.2) is a natural choice for solving this type of problems. The fuzzy logic controllers (FLCs) had been used by many researchers to solve the motion planning problem (Bagchi 1991, Beaufrere and Zeghloul 1995, Takeuchi et al. 1988, Martinez et al. 1994, Pin and Watanabe 1994). However, in all such studies, no effort was made to find an optimal FLC (instead an FLC is designed based on a particular user-specified membership function and rule base). Thus, the obtained collision-free path may not be optimal in any sense.

1.4.5 Robot Motion Planning Using Neural Networks

Lee and Bein (1990) used *neural network* as an optimizer to determine a collision-free trajectory for multiple robots. The positions or configurations of robots are taken as the variables of neural circuit and the energy of network is determined by combining various functions, in which one function is to make each robot approach to its goal and another helps each robot from not colliding with other robots or obstacles. Moreover, Chan et al. (1994) developed a neural network-based approach for planning collision-free motion of a convex shaped object in a planar workspace with static unconstrained shapes of obstacles. The motion planning problem is divided into two sub-problems, namely *find-space problem* and *find-path problem*. The find-space problem is to construct the configuration space of a given object and some obstacles and the find-path problem is to determine a collision-free

path from a given start location to a goal location for a point robot. Fig. 1.5 shows the schematic diagram of the proposed motion planning strategy. The find-space problem is solved by using a feed-forward neural network (FFNN) and the find-path problem is solved by using a bidirectional associative memory (BAM) which is similar to the Hopfield network.

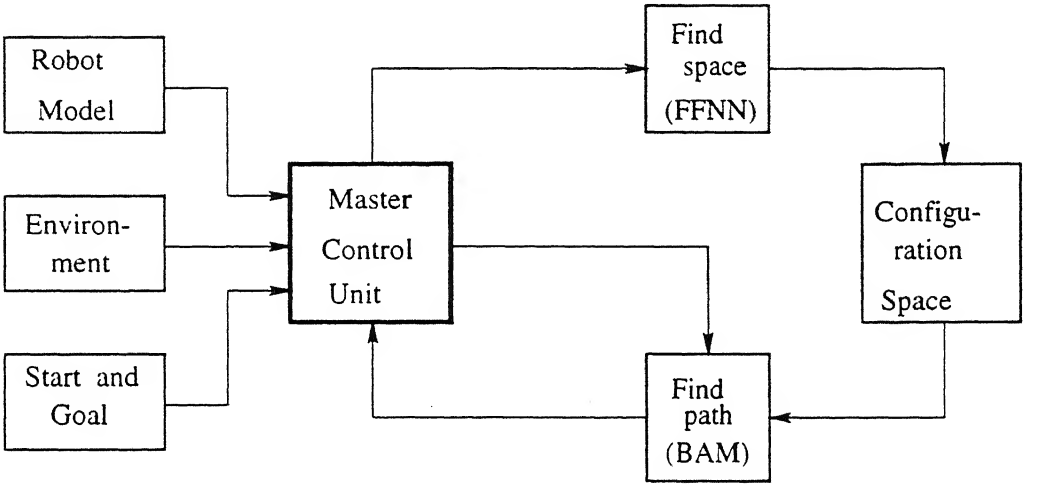


Figure 1.5: Motion planning strategy proposed by Chan et al. (1994).

1.4.6 Robot Motion Planning Using Genetic Algorithms

Mitchell (1988) used a *genetic algorithm*, a stochastic optimization approach, (refer to Section 2.3) along with a grid search as well as the visibility graph search methods to find the optimal collision-free path for the robot. However, the proposed technique was not very effective because of large computational complexities associated with grid search and graph search methods. Moreover, robot motion planning problems had also been solved by using genetic algorithm as an optimizer by Ahuactzin et al. (1992), Leung et al. (1994), Pinchard et al. (1995), and others.

1.4.7 Robot Motion Planning Using Learning Tools

Dorigo and Schnepf (1993) considered *genetic algorithm* as a learning tool in *behavior-based robotics* to develop an intelligent system. An autonomous agent should be able

to adapt its behavior to any changes in the environment. In behavior-based robotics, an attention is focussed on the design of appropriate behavioral modules and the coordination techniques to combine these behaviors in the most fruitful way so that the robot can navigate in a changing environment. In the proposed system, many classifier systems are running in parallel, as shown in Fig. 1.6. Two different learning activities are to be

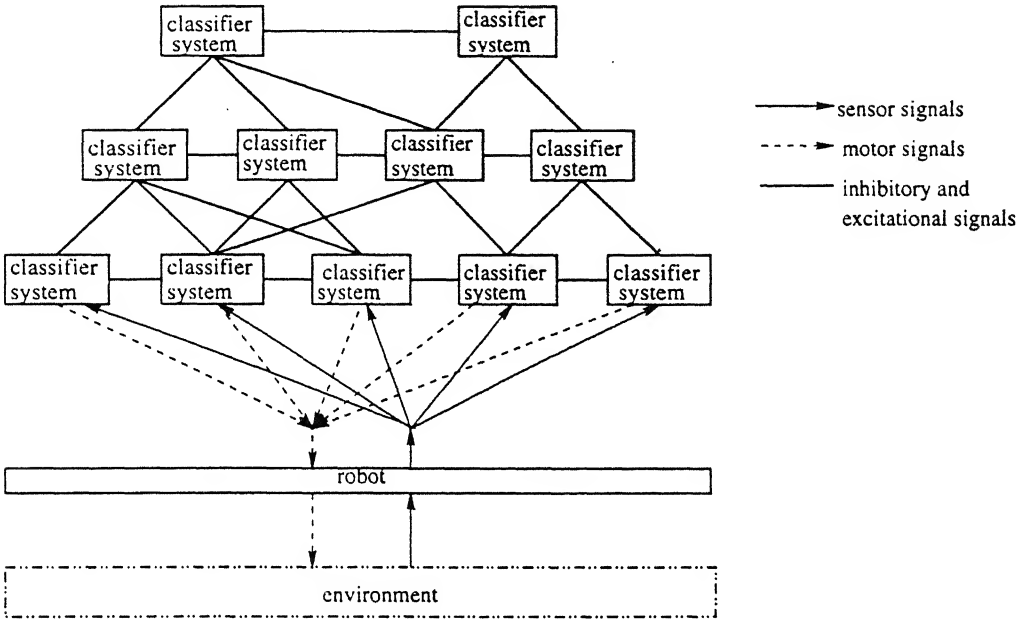


Figure 1.6: Learning technique in behavior-based robotics (Dorigo and Schnepf 1993).

performed, namely the learning of behavioral sequences and the learning of coordination sequences. The classifier systems at the lowest level have direct access to the environment via sensors and are used for learning behavioral sequences, whereas the classifier systems at higher levels learn to coordinate the activities of the classifier systems present at the lower level. Moreover, *evolutionary navigator* had been developed by Lin et al. (1994), Xiao et al. (1997) by using an evolutionary algorithm to solve the mobile robot navigation problem with moving obstacles. Besides this, some more work had been carried out to develop an intelligent motion planner by using different learning techniques such as *reinforcement learning* in which an agent can learn from success and failure, reward and punishment (Donnart and Meyer 1996, Millán 1996), *supervised learning* (Tani 1996), and others. The basic idea behind these learning techniques is that the planner should learn from the examples so that it can navigate in a dynamic environment with adequate reliability.

1.4.8 Gait Generation of Legged Robot

Muybridge (1899) studied the locomotion of animals by using successive photographs and his work is considered as a classic work in the study of walking gaits. Later on, the problem of human locomotion was also studied by him using the similar technique (Muybridge 1901). Moreover, the first mathematical model to analyze legged locomotion was developed by Tomovic and Karplus (1961). Hildebrand (1967) introduced the concept of *gait diagram* to describe the gaits of horses. The gait pattern can be either periodic or non-periodic.

A considerable amount of work had been devoted to the periodic gait generation problem of a legged vehicle. Both graphical as well as analytical approaches had been used to study the periodic gait generation. The periodic gait generation problem had been studied by many investigators. In this connection, work of Sindall (1964), McGhee and Frank (1968), Bessonov and Umnov (1973), Sun (1974), Rahman (1977), Song and Waldron (1987), Zhang and Song (1990) are important to mention. Some of the important results related to periodic gait established by previous researchers are stated here. McGhee and Frank (1968) used a nonlinear programming method and showed that a quadrupedal wave gait has the optimum stability among all quadrupedal periodic gaits in the range of $3/4 \leq \beta < 1$, where β is the duty factor (refer to Section 5.1.1). Later on, Rahman (1977) arrived at the same conclusion by using a linear programming method. Bessonov and Umnov (1973) studied the gait generation problem of a hexapod by using numerical methods and proved that a hexapodal wave gait has the optimum stability among all hexapodal periodic gaits in the range of $1/2 \leq \beta < 1$. Later on, Sun (1974) came to the same fact through his numerical studies of gaits. Moreover, Song and Waldron (1987) developed an analytical method for gait study and derived the *longitudinal gait stability margin* (s_L) of a wave gait. The longitudinal gait stability margin is defined as the shorter of the distances from the projection of the center of gravity to the front and rear boundaries along the axis of body movement (Fig. 1.7). A gait is statically stable if $s_L \geq 0$.

Non-periodic gaits are generally used in locomotion over a rough terrain. The first study reported in this area was conducted by Kugushev and Jaroshevskij (1975). Later on, several methods had been tried by various researchers to solve the problems of non-periodic gait generation of a legged robot. These work include the studies of free gaits (McGhee

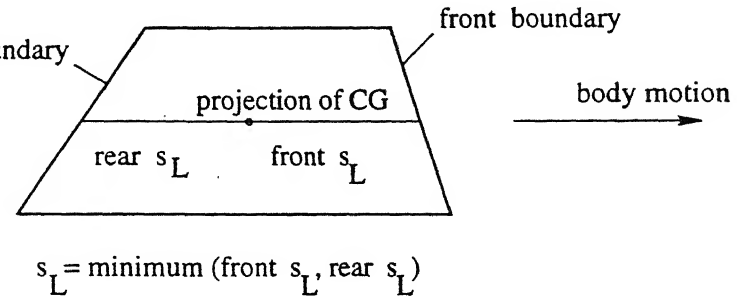


Figure 1.7: Longitudinal stability margin of a support pattern.

and Iswandhi 1979, Patterson et al. 1983, Kwak 1984, Pal and Jayarajan 1990, 1991); adaptive gaits (Hirose 1984, Kumar and Waldron 1989, Jiménez and Santos 1997); follow-the-leader gaits (Tsai 1983, Ozguner et al. 1984); the large-obstacle gaits (Wang 1983, Song 1984), and others. In all of these studies, either an analytical approach or a graphical approach had been used. Some of these work have been discussed here briefly. McGhee and Iswandhi (1979) proposed the free gait algorithm. The terrain was discretized into a number of cells which were classified into two groups, namely GO cells and NO-GO cells. The footholds were selected from the GO cells by maximizing the stability of the vehicle. The main disadvantage of their method was the high computational load. Moreover, the method did not perform well on uneven terrain. Kumar and Waldron (1989) developed an adaptive gait control algorithm in which a periodic gait will be generated by a legged robot while moving on smooth terrain, and a non-periodic gait pattern will be followed to adapt to uneven terrain. The terrain adaptive gait was generated by dynamically varying the parameters of an underlying wave gait. An accurate and deterministic geometric model of the terrain was considered in their approach. Pal and Jayarajan (1990, 1991) used a heuristic graph search technique for the generation of free gait of a legged vehicle. The main contribution of their work lies in forming the heuristic rules based on the support state transition table of an underlying fixed gait. The support state of a leg was expressed as a binary number (1 if the leg is on the ground and 0 if the leg is in the air). For a six-legged robot, there are 2^6 or 64 possible support states and out of those, only a few can ensure stability of the vehicle. A support state transition table was formed based on a certain optimization criterion and the A* algorithm was used for generating an optimal route to the goal. Their method requires large computer memory. Moreover, as the criterion of optimization changes, a separate support state transition table has to be formed. Moreover, the generated gait may be only locally optimal. Shih and Klein (1993)

proposed a heuristic approach to study the legged locomotion over rough terrain. In their algorithm, there were three levels of decision making, namely adaptation in body motion, adaptation in leg sequence and adaptation in foot positioning. Their algorithm was able to generate free gait for the legged robot while moving on rough terrains with depressions, whereas on a smooth terrain, it used to generate a periodic gait. Moreover, to deal with uncertainties and model imprecision with increased flexibility and robustness, *behavior-based* gait generation technique was suggested by Brooks (1989), Wettergreen et al. (1995). In their approaches, global behavior was constructed on-line from combinations of reactive elemental behaviors.

1.4.9 Drawbacks of the Traditional Methods of Gait Generation

The traditional methods of gait generation which include both graphical as well as analytical approaches have the following limitations:

1. Gait generated by the conventional methods is far from being optimal in any sense. The generated gait may be only locally optimal.
2. In optimal gait generation problems, some of the variables may be discrete in nature and handling of these variables becomes difficult in the traditional optimization techniques.
3. The traditional methods (particularly the graphical methods) are computationally expensive.

1.4.10 Gait Generation Using Soft Computing Techniques

Soft computing (discussed in Chapter 2) techniques include fuzzy logic (FL), neural network (NN), genetic algorithm (GA), and others. The gait generation problems of a legged robot had also been tackled by many researchers by using the soft computing techniques. Min and Bien (1992) solved the foothold selection problem of a quadruped by using a neural network. The desired footholds were taken as the output variables of the neural circuit and differential equations which tend to minimize the energy of the network were derived. Moreover, a distributed neural network controller had been used to control a

hexapod robot by Chiel et al. (1992). The main drawback of using neural network lies in the fact that there is a chance of the solution for getting trapped into local minima. Moreover, Garis (1990) used a genetic algorithm to design the neural network and it was used for controlling a bipedal walking machine. He introduced the concept of *behavioral memory* to reduce the length of the search. Unfortunately, his algorithm was not suitable for on-line implementation with a real robot as it requires a large number of evaluations. Later on, Lewis et al. (1992) extended the idea of behavioral memory and developed a combined GA-NN technique for control of a six-legged robot. In their algorithm, the optimal weights of the neural network were determined by using a GA. More recently, Gruau and Quatramaran (1996) has proposed an algorithm in which evolutionary algorithm (EA) has been used to evolve an artificial neural network (ANN) which can control the locomotion of an eight-legged robot. Moreover, Magdalena and Velasco (1996) developed a combined GA-FL technique and it was applied to a gait synthesis problem of a biped robot. Here, the purpose of using the FLC was to define joint trajectories in such a way that the biped system describes a regular walk without falling or stopping. Here, the GAs were used not with an optimization aim, but with a diversification aim. The knowledge base of an FLC was first determined suitable for controlling a regular walk with a certain speed and stride length (refer to Section 5.1.1), and then GAs were used to create other knowledge bases capable of controlling regular walks with different speeds and stride lengths.

1.4.11 Combined Path and Gait Generation of Legged Robots

Despite such a volume of study on individual tasks, a little amount of work had been carried out to solve the problem of combined path and gait generation of a legged robot. Lee and Song (1991) studied the locomotion of a four-legged walking machine in an obstacle-strewn environment. In their algorithm, the obstacle-free path was generated first based on a Bezier curve approximation and after that the gait was generated so that the vehicle can follow the path and maintain its stability. Thus, in their algorithm, the problem of combined path and gait generation had been solved in stages. In practice, both path and gait generations are to be done simultaneously. Moreover, their approach might not be suitable for on-line implementation due to high computational load. Moreover, a discrete formulation of the motion planning problem for statically stable walking vehicles was proposed by Chen and Kumar (1996). In their work, only the gait generation problem

(that includes the sequencing of legs and placing of the feet on appropriate locations on terrain) had been discussed in detail. Thus, their approach was able to yield only a partial solution to the complex problem of combined path and gait generation of a legged robot. Thus, determining a computationally faster algorithm to solve this complicated task, in an optimal sense, is an issue of utmost importance and is the focus of this study.

1.5 Objective and Scope of Present Work

In the present work, an attempt has been made to develop some algorithms by combining the two techniques, namely genetic algorithm (GA) and fuzzy logic (FL) and these algorithms have been used to solve the problems of path generation and gait generation of a legged robot. The main purpose of this study is to develop a computationally faster algorithm which can be implemented on-line, to tackle the problem of combined path and gait generations of a legged robot. The objective of this study can be listed as follows:

1. **An attempt to develop methods for controlling multi-legged robots:** While navigating, a multi-legged robot will have to plan both the path as well as gait simultaneously. Thus, an autonomous and intelligent legged robot should be able to plan its path and gait in a changing environment. In this work, the issues of path planning and gait planning have been studied, at first, separately and after that the problems of combined path and gait generations have also been studied, in detail.

   A *fuzzy-genetic algorithm* has been developed in which a fuzzy logic technique is used to improve the performance of a genetic algorithm. The time-optimal path planning problems of a mobile robot in the presence of stationary obstacles are solved by using the proposed fuzzy-genetic algorithm. Moreover, the performance of the proposed fuzzy-genetic algorithm has been compared with that of the *tangent graph technique (along with A* algorithm)* and the *steepest descent method with a penalty function approach* separately.

   A *genetic-fuzzy system* has been proposed in which the performance of a fuzzy logic controller has been improved by using a genetic algorithm. Here, an optimized fuzzy logic controller (FLC) is found by using a genetic algorithm and has been used to solve the navigation problems of a mobile robot in the presence of moving obstacles.

In this study, several methods of tuning a fuzzy logic controller by using a GA have been studied in detail and their performances have been compared.

The similar genetic-fuzzy system has also been used to solve the gait generation problem of a multi-legged robot, in an optimal sense. The problems of crab gait generation while crossing a ditch and turning gait generation have been solved separately with the help of the proposed genetic-fuzzy system. Moreover, a comparative study has been made regarding the performances of the proposed genetic-fuzzy system and the manually-constructed fuzzy logic controllers (both used for solving the gait generation problems).

An attempt has also been made to solve the problem of combined path and gait generations by using the genetic-fuzzy system. In this study, a six-legged robot will have to plan its path and gait simultaneously, in an optimal sense. It is a complicated task and no single traditional approach had solved it completely, in the past. This problem of combined path and gait generations has also been solved successfully by using the proposed genetic-fuzzy system. The performance of the proposed genetic-fuzzy system has been compared with that of the manually-constructed fuzzy logic controllers.

2. **Fast and efficient computation using GA-Fuzzy approach:** In this study, an attempt has been made to develop a fast and efficient technique which can tackle the path planning and gait planning problems in an optimal way. As an FLC is computationally less expensive, the computational load of the proposed genetic-fuzzy system is bound to be less than that of the traditional approaches. The execution time of the GA-tuned FLCs has been determined in a HP 9000/K200 machine, while solving a complicated task, such as simultaneous path and gait planning of a six-legged robot and it is found to be quite low. Thus, this algorithm can be implemented on-line, to solve the problem of combined path and gait planning of a multi-legged robot.

3. **Special situation - automatic rule generation for an FLC using a GA:** Although, an FLC is an effective tool for dealing with imprecision and uncertainty, determining an appropriate knowledge base of an FLC is a difficult task. It is important to mention that the performance of an FLC depends mainly on its knowledge base. In this study, a new approach has been proposed for determining a good rule base of an FLC by using a GA. In this approach, a GA will discover rules for an

FLC by optimally choosing an output linguistic for input linguistic combinations. Thus, a method for two-stage automatic design of an FLC has been developed by using a GA. This method is called *GA-based tuning of a GA-learned rule base*. It is a flexible approach and has been successfully implemented to solve the navigation problems of a mobile robot among moving obstacles.

However, the present work does not focus on the following issues:

1. The proposed algorithms based on GA-Fuzzy combinations are suitable for flat terrain only. In practice, the terrain may be uneven and possibly with some slope and the algorithms are to be modified accordingly.

2. The effectiveness of the proposed algorithms is tested through computer simulations for a six-legged robot only. These algorithms are to be modified accordingly to tackle the navigation problem of other type of multi-legged robots, namely four-legged, eight-legged robots.

3. In the present work, the method of automatic fuzzy rule generation using a GA (GA-based tuning of a GA-learned rule base) has been implemented only for the path planning problems of a mobile robot. It has not yet been implemented for the gait generation problem of a legged robot.

4. In this study, a six-legged robot will have to plan its path and gait simultaneously, in the optimal sense. This problem of combined path and gait generations can be treated as a multi-objective optimization problem and it can be solved by using multi-objective GA to find multiple trade-off solutions.

5. In the present work, an attention is focused only on the static stability of the vehicle and the issues related to the dynamic stability of the vehicle are not considered.

6. In this work, the effectivenes of the proposed GA-Fuzzy approaches has been tested only through computer simulations and no experiment has been carried out with real robots to validate the proposed algorithm.

1.6 Summary

This chapter provides a survey on the different traditional methods (both graphical as well as analytical) of path and gait planning of a robot. The main drawback of these methods is their high computational load. Moreover, each of these methods is suitable for solving a particular type of problems. The problem of combined path and gait generation of a legged robot has received a little attention, till now. It is a difficult task and no single traditional approach is able to provide a complete solution. Soft Computing techniques have also been used by many investigators to solve the path generation and gait generation problems of a robot. Thus, developing a computationally tractable and versatile algorithm to solve this complicated task of combined path and gait generation of a legged robot is a subject for future research.

1.7 Overview of the Thesis

This thesis is divided into nine chapters. Chapter 2 includes a brief introduction to Soft Computing. Moreover, the working principles of a fuzzy logic controller and genetic algorithms have been discussed in Chapter 2. The working principle of a fuzzy-genetic algorithm has been explained and its effectiveness is tested on a number of navigation problems of a mobile robot among static obstacles in Chapter 3. Chapter 4 introduces a genetic-fuzzy system to solve the navigation problems of a mobile robot among moving obstacles. An algorithm has been developed and tested for periodic gait generation of a six-legged robot in Chapter 5. The genetic-fuzzy system is used to generate optimal/near-optimal crab gait while crossing a ditch and turning gait of a six-legged robot as discussed in Chapters 6 and 7, respectively. The problems of combined path and gait generation of a six-legged robot have been solved using the similar genetic-fuzzy system in Chapter 8. Finally, Chapter 9 ends with some concluding remarks and suggests the scope for future work.

Chapter 2

SOFT COMPUTING

A brief introduction is given to Soft Computing, in this chapter, which is used to solve the complicated real-world problems. Soft Computing includes fuzzy logic (FL) technique, neural networks (NNs), genetic algorithms (GAs), and others. The working principles of a fuzzy logic controller (FLC) and genetic algorithms (both binary-coded GA as well as real-coded GA) have been explained in this chapter. Moreover, it provides a detailed survey on the combined GA-Fuzzy approaches proposed by several investigators.

2.1 Introduction to Soft Computing

Classical reasoning and modeling approaches, which are based on boolean logic, analytical models, crisp classifications, and deterministic search will almost surely fail to solve real-world and often ambiguous problems because these problems are ill-defined, difficult to model and exhibit large-scale solution spaces. The notion of *soft computing* has emerged recently which accommodates imprecision and uncertainty to allow reasoning and computation usually needed for practical applications. The term "*soft computing*" was introduced by L. A. Zadeh (1992), in which precision is traded for tractability, robustness, ease of implementation, and lower solution cost. The main components of soft computing are fuzzy logic (FL), neural networks (NNs), genetic algorithms (GAs), and others. The fuzzy logic technique is a powerful tool for dealing with imprecision and uncertainty (Kosko 1994). Moreover, it is an effective tool for local search. Artificial neural networks are generally used for learning and adaptation (Kosko 1994). Genetic algorithms

are population-based search and optimization techniques which mimic the mechanics of natural selection and natural genetics (Goldberg 1989). It is also used as a key element in many learning techniques (Dorigo and Schnepf 1993). Many researchers developed hybrid systems in order to get advantages of each methodology and to overcome their individual limitations. Fuzzy logic technique has been merged with neural networks to develop a combined FL-NN approach by several investigators. In this connection, the work of Takagi and Hayashi (1991), Enbutsu et al. (1991), Vestli et al. (1993), Li (1997) are worth mentioning. Similarly, the combined FL-GA approaches are developed by several researchers. Cordon et al. (1997) provide an extensive survey on this topic. The need of a combined GA-NN approach is also realized by many investigators. In this connection, the work of Harp and Samad (1991), Harvey et al. (1993) are significant. Most recently, all the three main components of soft computing have also been merged to develop a GA-FL-NN approach for solving real-world problems by Ishigami et al. (1995).

Here we discuss two of these three main components of Soft Computing - fuzzy logic technique and genetic algorithms, primarily because these two techniques have been used in this thesis. Interested reader may refer to Kosko (1994) for NN details.

2.2 Introduction to Fuzzy Concept and Fuzzy Logic Controller

In practice, we try to maximize the usefulness of a model, while constructing it. Usefulness of a model depends on the interactions among three key characteristics, namely *complexity*, *credibility*, and *uncertainty* (Klir and Yuan 1997). Out of these three characteristics, *uncertainty* plays the most important role in maximizing the usefulness of system models. In general, allowing more uncertainty tends to reduce complexity and increase credibility of the resulting model. Before Zadeh's seminal paper (Zadeh 1965), the probability theory which works based on Aristotelian two-valued logic, was the sole agent for dealing with *uncertainty*. The concept of fuzzy set theory (a new concept for dealing with uncertainty) was first conceived by L. A. Zadeh, in the year 1965. His paper was a challenge against the probability theory. He argued that probability theory is capable of representing only one of several distinct types of uncertainty. Fuzzy sets are defined as sets with boundaries that are not precise. Fuzzy set is different from classical or crisp set. In crisp set, the individuals in a given universe of discourse are clearly divided into two groups, namely members and

non-members. Thus, in crisp set, the boundaries are clearly defined. On the other hand, a fuzzy set can be defined mathematically by assigning to each individual in the universe of discourse a value representing its grade of membership (degree of belongingness). The membership value (real number) may vary from 0 to 1 and there are, in fact, several types of membership function distributions in use. The most broadly used parameterized membership functions are: triangular, trapezoidal, Gaussian, bell and sigmoidal. It is interesting to note that if the membership grade is high, the element belongs to the class with a higher degree of certainty. In the fuzzy set, full membership and non-membership are indicated by 1 and 0, respectively. Thus, a crisp set is considered as a special case of the more general concept of a fuzzy set.

The membership function of a fuzzy set A is denoted by μ_A, that is,

$$\mu_A : X \rightarrow [0, 1]$$

where X denotes the universe of discourse or universal set.

2.2.1 Some Standard Fuzzy Operations

- **Complement of a fuzzy set:**
  The complement, $\bar{A}$, of fuzzy set A with respect to the universal set X is defined for all $x \in X$ as

$$\bar{A}(x) = 1 - A(x).$$

- **Intersection of fuzzy sets:**
  Intersection of two fuzzy sets, A and B, is defined for all $x \in X$ as

$$(A \cap B)(x) = \min[\mu_A(x), \mu_B(x)].$$

  It is interesting to note that intersection is analogous to the logical AND (conjunction) operation.

- **Union of fuzzy sets:**
  Union of two fuzzy sets, A and B, is defined for all $x \in X$ as

$$(A \cup B)(x) = \max[\mu_A(x), \mu_B(x)].$$

  It is to be noted that union operation corresponds to the logical OR (disjunction) operation.

Interested reader may refer to Klir and Yuan (1997) for other fuzzy operations.

2.2.2 Working Principle of a Fuzzy Logic Controller

Fuzzy logic controller (FLC), a successful application of the fuzzy set theory, is a powerful tool for dealing with imprecision and uncertainty (Mamdani and Assilian 1975). An FLC tries to model the behavior of a human operator who would be able to control a process and it does not depend on the mathematical description of the process. An FLC can be used to control a complicated process which is difficult to model mathematically or the mathematical model is severely nonlinear. Here, human knowledge is expressed in terms of linguistic control rules.

An FLC consists of four modules, namely a fuzzy rule base, a fuzzy inference engine, fuzzification, and De-fuzzification. Fig. 2.1 shows a schematic diagram explaining the working cycle of an FLC.

The following steps are involved in the working cycle of an FLC:

- The variables (condition and action) needed to control a particular process are chosen and measurements are taken of all the condition variables.
- The measurements taken in the previous step are converted into appropriate fuzzy sets to express measurement uncertainties. This is known as *fuzzification.*
- The fuzzified measurements are then used by the inference engine to evaluate the control rules stored in the fuzzy rule base and a fuzzified output is determined.

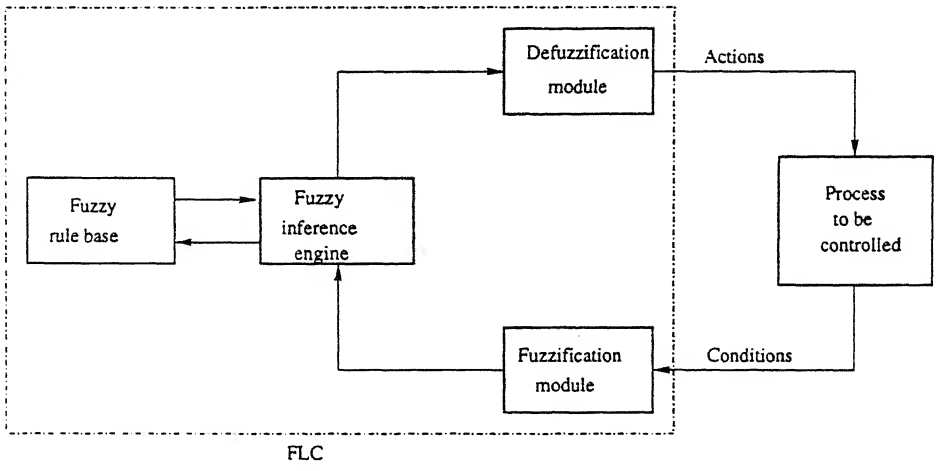


Figure 2.1: A schematic showing the working cycle of an FLC.

- The fuzzified output is then converted into a single (crisp) value. This conversion is called a *De-fuzzification.* The De-fuzzified values represent actions to be taken by the FLC in controlling the process.

The working principle of an FLC is explained below with the help of an example (refer to Fig. 2.2). For simplicity, we assume that there are the following two fuzzy control rules:

RULE 1: IF s_1 is A_1 and s_2 is B_1 THEN f is C_1
RULE 2: IF s_1 is A_2 and s_2 is B_2 THEN f is C_2.

If s_1^* and s_2^* are the inputs for fuzzy variables s_1 and s_2 and if μ_{A1} and μ_{B1} are the membership functions for A and B, respectively, then the grade of membership of s_1^* in A_1 and the grade of membership of s_2^* in B_1 are represented by $\mu_{A1}(s_1^*)$ and $\mu_{B1}(s_2^*)$, respectively for rule 1. Similarly, for rule 2, $\mu_{A2}(s_1^*)$ and $\mu_{B2}(s_2^*)$, are used for the grades of membership. The firing strengths of the first and second rules are calculated as follows:

$$\alpha_1 = \min\left(\mu_{A1}(s_1^*),\ \mu_{B1}(s_2^*)\right), \tag{2.1}$$

$$\alpha_2 = \min\left(\mu_{A2}(s_1^*),\ \mu_{B2}(s_2^*)\right). \tag{2.2}$$

The membership function of the combined control action C is given by

$$\mu_C(f) = \max\left(\mu_{C1}^*(f),\ \mu_{C2}^*(f)\right). \tag{2.3}$$

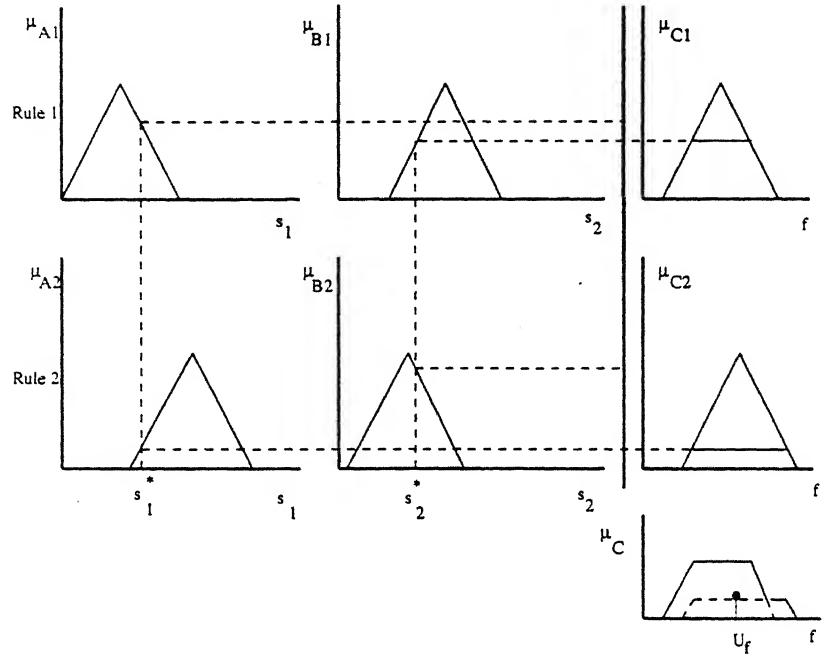


Figure 2.2: A schematic showing the working principle of an FLC.

The *center of area method* is employed for De-fuzzification and the method can be represented by

$$U_f = \frac{\sum_{j=1}^{p} A(\alpha_j) \times f_j}{\sum_{j=1}^{p} A(\alpha_j)}, \tag{2.4}$$

where U_f is the output of the controller, $A(\alpha_j)$ represents the firing area of the j-th rule, p is the total number of fired rules and f_j represents the centroid of the area. It is to be mentioned that there exists some other techniques of De-fuzzification also, namely center of maxima method, mean of maxima method (Klir and Yuan 1997).

2.3 Introduction to Genetic Algorithms

The concept of Genetic Algorithms (GAs) was introduced by Prof. John Holland of the University of Michigan, Ann Arbor, USA, in the year 1965, although his seminal book appeared in the year 1975 (Holland 1975). GAs are population-based search and optimization techniques which work using Darwin's principle of natural selection (Goldberg 1989). GAs are very different from classical search and optimization methods. Classical search and optimization methods can be classified into two distinct groups, namely di-

rect method and gradient-based method (Deb 1995). The derivative information is not required in direct method, whereas it is a necessity for gradient-based method. In addition, there are some difficulties with most of the traditional methods of optimization, as discussed below (Deb 1998):

- Convergence to an optimal solution depends on the chosen initial solution.
- Most of the algorithms get stuck to a sub-optimal solution.
- An algorithm efficient in solving one optimization problem may not be efficient in solving a different problem.
- Algorithms are not efficient in handling problems having discrete variables.
- Algorithms can not be efficiently used on a parallel machine.

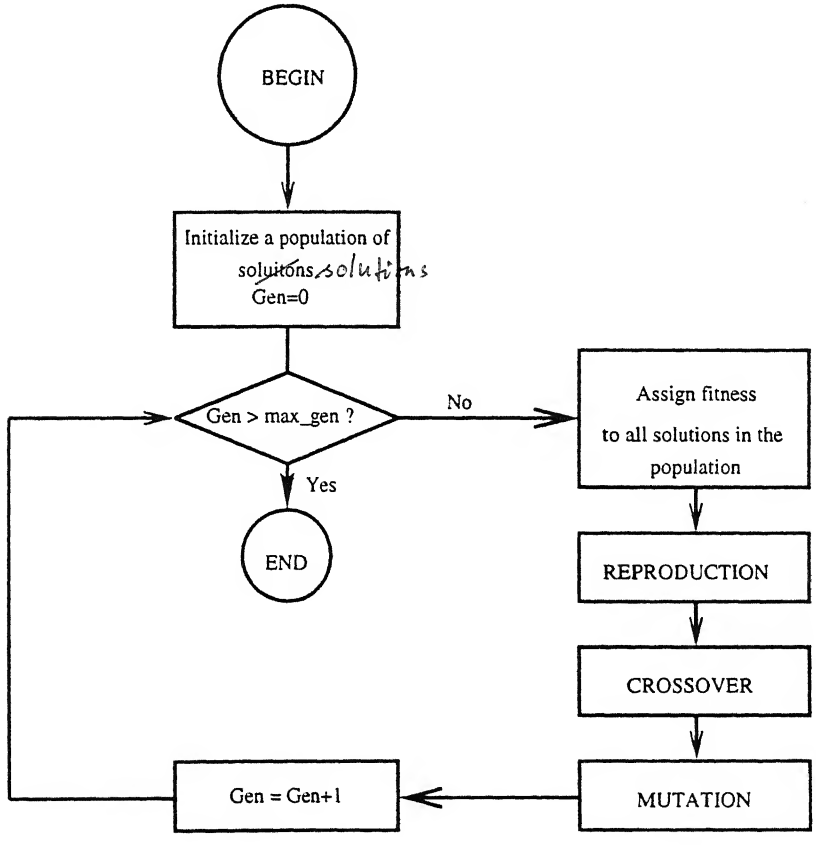


Figure 2.3: A schematic showing the working cycle of a GA.

It is interesting to note that a GA can overcome almost all of the difficulties faced by the classical methods of optimization. The working cycle of a GA is shown in Fig. 2.3, in the form of a flowchart. The operation of a GA begins with a population of initial solutions, chosen at random. Thereafter, the fitness value (objective function value) of each member (solution) in the population is calculated. The population is then operated by three main operators, namely reproduction, crossover, and mutation, to create a new population. The new population is further evaluated and tested for termination. It is important to mention that there could be different termination criteria also, namely a maximum number of generations, a desired accuracy in the solution, and others. There exist several versions of the GA, such as binary-coded GA, real-coded GA, messy GA, and others. The working principle of a binary-coded GA and a real-coded GA have been discussed briefly in the following.

2.3.1 Binary-coded Genetic Algorithms

In a binary-coded GA, all problem variables are coded in finite-length binary strings. For example, three variables x_1, x_2, and x_3 can be represented by 4, 3, and 5 bit substrings as follows:

$$\underbrace{0111}_{x_1}\ \underbrace{101}_{x_2}\ \underbrace{00110}_{x_3}$$

The total string length (l) of a solution is then 12. The length of each substring is determined by the required accuracy in each variable. In order to retrieve the corresponding variable values, the following decoding scheme is usually used:

$$x_i = x_i^L + \frac{x_i^U - x_i^L}{2^l - 1} \times d, \tag{2.5}$$

where x^L and x^U are the lower and upper bounds of the variable x, respectively; d indicates decoded value of the string. Once the values of the variables are known, objective function value can be determined. For maximization problems, a string's fitness can be equal to the objective function value. However, for minimization problems, the fitness can be calculated as the reciprocal of the objective function value so that solutions with smaller objective function value get larger fitness. Thus, for minimization problems, fitness is usually calculated as follows:

$$Fitness = \frac{1}{[1 + f(x_1, x_2, x_3)]}. \tag{2.6}$$

It is important to note that a binary-coded GA can be used to solve a variety of problems by only changing the definition of coding a string.

Reproduction operator selects good strings in a population using fitness information and forms a *mating pool.* There exists a number of reproduction schemes in the GA literature, namely proportionate selection, ranking selection, tournament selection, and others (Goldberg and Deb 1991). Out of all these schemes, tournament selection scheme is the most popular due to its simplicity and controlled takeover property. In a binary tournament selection, two strings are chosen at random from the population and the best string is selected and copied in an intermediate population, called the mating pool. This process continues by comparing two strings at a time till the mating pool has the same size as the original population size. Thus, this operator emphasizes the good strings of a population and makes duplicate copies of them in a mating pool.

In the crossover, new strings are created by exchanging information among strings of the mating pool. There are several types of crossover operators in the GA literature, namely single-point crossover, two-point crossover, uniform crossover, and others (Spears and De Jong 1991). In a single-point crossover, two strings are chosen from the mating pool, and also a crossing site is chosen along the string. Thereafter, all bits on the right side of the crossing site are exchanged between both the strings, as illustrated in the following:

Parent 1	00	011		00	100	Child 1
			$\Rightarrow$			
Parent 2	11	100		11	011	Child 2

This operator allows partial information to be exchanged between two good strings found using the reproduction operator. Crossover operator is mainly responsible for the search of new strings. In order to reduce the chance of destructing already found good strings, crossover is usually performed with a probability, p_c slightly smaller than one.

Mutation means a sudden change of a parameter. In a binary-coded GA, mutation operator changes 1 to 0 and vice versa with a small probability, p_m, as follows:

$$00000 \Rightarrow 00100$$

In the above example, the third bit has changed its value, thereby creating a new solution.

Mutation is used for achieving a local change around the current solution. In short, reproduction operator selects good strings and crossover operator recombines two good strings to hopefully create better strings. The mutation operator alters a string locally to create a new string.

After reproduction, crossover, and mutation are applied to the whole population, one generation of a GA is completed. The operation of a GA is very similar to the evolutionary principle. If good strings are created by crossover and mutation operators, reproduction emphasizes them and they have more chance of getting mated with other good solutions. On the other hand, if crossover and mutation create bad strings, reproduction ruthlessly eliminates them from further processing. The string copying and substring exchange operations in GAs may seem at first to be random operations, a careful thought will prevail that although random numbers are used extensively in a GA, the search is not a random search. Instead, the randomness in the search operators provide the necessary stochasticities for a GA not to get stuck at suboptimal solutions.

2.3.2 Real-coded Genetic Algorithms

In solving optimization problems having continuous search space, a number of difficulties are faced while using a binary-coded GA. One difficulty is the *hamming cliffs* associated with certain strings, for which a transition to a neighboring solution requires alteration of many bits. Hamming cliffs in the binary coding cause artificial hindrance to a gradual search in the continuous search space. The other difficulty is its inability to achieve any arbitrary precision in the optimal solution. In binary-coded GAs, the string length must be chosen *a priori* to enable GAs to achieve a certain precision in the solution. The more the required precision, the larger is the string length. For large strings, the population size requirement is also large (Goldberg et al. 1992), thereby increasing the computational complexity of the algorithm.

To overcome these difficulties, real-coded GAs have been developed by several investigators (Wright 1991, Eshelman and Schaffer 1993, Deb and Agrawal 1995, Radcliffe 1991, Michalewicz 1994) for solving optimization problems in continuous search space. Here, a real-coded GA, developed by Deb and Agrawal (1995) has been discussed in detail.

A tournament selection scheme is used as a reproduction operator, where two solutions are compared and the best (in terms of fitness value) is selected to form a mating pool.

A special type of crossover operator known as *Simulated Binary Crossover (SBX)* is used for continuous search space in which the search power is expressed in terms of the probability distribution of any arbitrary child to be created from any two given parents. In order to define the spread of the children points with respect to that of the parent, spread factor α is defined as follows:

$$\alpha = \left| \frac{Ch_1 - Ch_2}{Pr_1 - Pr_2} \right|, \tag{2.7}$$

where Ch_1, Ch_2 are the children points and Pr_1, Pr_2 are the parent points. Depending on the value of the α, the crossover has been classified into three groups, namely *contracting crossover* ($\alpha < 1$), *expanding crossover* ($\alpha > 1$), and *stationary crossover* ($\alpha = 1$). The probability distributions of contracting and expanding crossover are shown in Fig. 2.4. It is to be noted that the cumulative probability of all the possible states is equal to one.

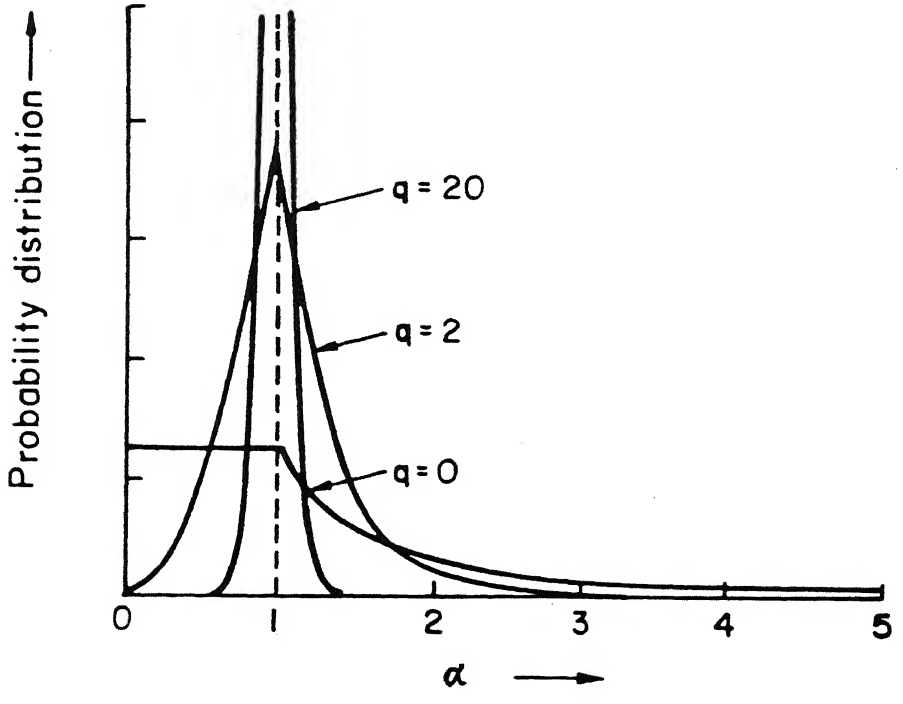


Figure 2.4: Probability distributions of contracting and expanding crossover.

For contracting crossover, the probability distribution is taken as follows:

$$C(\alpha) = 0.5 \times (q+1)\alpha^q, \tag{2.8}$$

where q is any nonnegative real number. So, the commutative probability at $\alpha = 1$, equals to 0.5. On the other hand, the probability distribution for expanding crossover is considered as follows:

$$Ex(\alpha) = 0.5 \times (q+1)\frac{1}{\alpha^{q+2}}. \tag{2.9}$$

It is interesting to note that the probability of all expanding crossovers is found to be 0.5, equal to that of the contracting crossovers and the distribution largely depends on the exponent q. For small values of q, points far away from the parents are likely to be chosen, whereas for large values of q, only points closer to the parents are likely to be chosen. In order to create two children solutions Ch_1 and Ch_2 from the parents Pr_1 and Pr_2 using the above probability distribution, the following procedure is used:

- Create a random number u between 0 and 1.
- Find a $\acute{\alpha}$ for which the cumulative probability

$$\int_0^{\acute{\alpha}} p(\alpha)d\alpha = u, \tag{2.10}$$

  where $p(\alpha) = C(\alpha) or Ex(\alpha)$.

- Knowing the value of $\acute{\alpha}$, the children points are calculated as follows:

$$Ch_1 = 0.5 \times [(Pr_1 + Pr_2) - \acute{\alpha}\,|Pr_2 - Pr_1|], \tag{2.11}$$

$$Ch_2 = 0.5 \times [(Pr_1 + Pr_2) + \acute{\alpha}\,|Pr_2 - Pr_1|]. \tag{2.12}$$

Although this crossover operator has the power to create any solution in the real space with varying importance, this can be modified to create solutions only between two fixed bounds (x^L,x^U). The procedure is as follows: The probability distribution function shown in Fig. 2.4, is multiplied by a suitable factor depending on these limits and the location of the parent solutions, so as to make a zero probability of creating any solution outside these limits. For the child solution closer to parent Pr_1, this factor can be computed as $\frac{1}{1-\bar{\gamma}}$, where $\bar{\gamma}$ is the cumulative probability of creating solutions from $x = -\infty$ to $x = x^L$. Similarly, a factor for the child solution closer to Pr_2 can also be calculated.

The following procedure is adopted to create a mutated value:

- Create a random number, u between 0 and 1.
- Calculate the perturbance factor $\bar{\delta}$, corresponding to u using the following equations:

$$\bar{\delta} = \begin{cases} (2u)^{\frac{1}{q+1}} - 1 & \text{if } u < 0.5 \\ 1 - [2(1-u)]^{\frac{1}{q+1}} & \text{if } u \geq 0.5. \end{cases} \tag{2.13}$$

- The mutated value is calculated as follows:

$$Ch = Pr + \bar{\delta} \times \delta_{max}, \tag{2.14}$$

where δ_{max} is the maximum perturbance between the parent Pr and child Ch.

After reproduction, crossover and mutation are applied to the population, one cycle of a GA is completed.

The performance of a GA largely depends on the selection of its parameters, namely population size, coding representation of decision variables (in case of a binary-coded GA), crossover and mutation probabilities. Thus, to solve a problem efficiently with a GA, the user must be aware of the studies related to appropriate parameter setting (Deb and Agrawal 1999). The GA-operators (crossover and mutation) are generic in nature, so a GA can be used to solve a wide variety of problems. It is important to note that no gradient information is required in the working of a GA. Since no gradient information is used, a GA may require comparatively more function evaluations than classical search and optimization methods in solving simple, differentiable, unimodal functions. Therefore, it may not be advantageous to apply a GA to such simple functions.

2.4 Introduction to GA-Fuzzy Combinations

Genetic algorithms (GAs) are adaptive computational procedures modeled on the mechanics of natural genetic systems. GAs are powerful tools for optimization and they are used as learning tools for solving many real-world complex problems. GAs require only function value information and work on binary variables (in case of binary-coded GAs). On the other hand, fuzzy logic controller (FLC) is computationally faster and can handle uncertainty and imprecision. Moreover, it is an efficient tool for local search. It is important to note that the performance of an FLC depends on its knowledge base and determination of a good knowledge base is a difficult task.

To get advantages of both the techniques, the approaches based on GA-Fuzzy combinations have been developed by several investigators (Cordon et al. 1997). Research in this area is going on in both the directions - in one approach, an FLC is used to improve the performance of a GA, whereas in the other implementation, a GA is used to improve the performance of an FLC. Both the approaches have been discussed below in detail.

2.4.1 Approach 1 (Application of FLC to improve the performance of a GA):

A considerable amount of work had been carried out, in the past, by several researchers. Lee and Takagi (1993a) proposed a dynamic parametric GA (DPGA) in which FLCs are used for controlling GA parameters. They used three FLCs for this purpose. The performance was measured in terms of *phenotypic diversity measures* (PDM), as defined below:

$$PDM_1 = \frac{f_{best}}{\bar{f}}, \tag{2.15}$$

$$PDM_2 = \frac{\bar{f}}{f_{worst}}. \tag{2.16}$$

where f_{best}, $\bar{f}$, f_{worst} represent the best fitness, the average fitness and the worst fitness, respectively. It is important to note that both PDM_1 and PDM_2 lie in the range (0,1). If they are near to 1, convergence has been reached, whereas if they are near to 0, the population shows a high level of diversity. The changes in the best fitness values since the last control action are taken as inputs of the FLCs and the outputs of the three FLCs are variables that control variations in the current mutation probability, crossover probability, and population size, respectively. Fig. 2.5 shows the schematic diagram of a DPGA. They

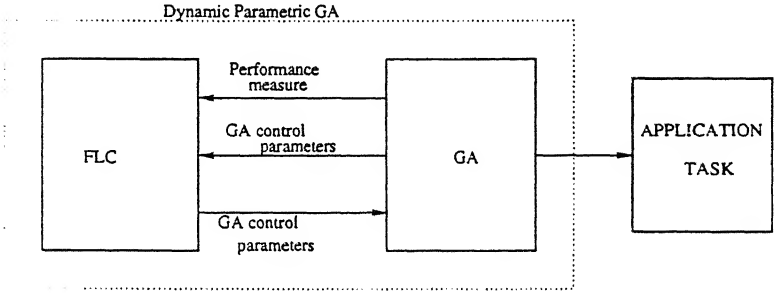


Figure 2.5: A schematic showing the working principle of a DPGA.

used DPGA for controlling an inverted pendulum and showed that the performance of a DPGA is better than that of a static GA.

Xu and Vukovich (1993), Xu et al. (1994) developed fuzzy GAs (FGA) which

- choose control parameters before a GA is run,
- adjust the control parameters on-line to dynamically adapt to new situations, and
- assist the user in accessing, designing, implementing and validating the GA for a given task.

They used two FLCs. The inputs of the FLCs were maximum number of generation and population size and the outputs were p_c and p_m. The FGAs were found to be more efficient than standard GA in solving the traveling salesman and other optimization problems.

Herrera et al. (1994) used genotypic diversity measures (GDMs) for describing the state of the population in a real-coded GA. They introduced the *variance average of the*

chromosomes (VAC) and the *average variance of the alleles* (AVA) for the purpose of diversity measures. The VAC and AVA are defined as follows:

$$VAC = \frac{\sum_{i=1}^{N}(\bar{S}_i - \bar{S})^2}{N}, \tag{2.17}$$

$$AVA = \frac{\sum_{j=1}^{N}\sum_{i=1}^{L}(S_{ij} - \bar{S}_j)^2}{LN}. \tag{2.18}$$

where S_i, $i = 1, \ldots, N$, denotes the chromosome i; S_j, $j = 1, \ldots, L$, indicates the vector of genes with position j in the population; S_{ij} denotes the gene with position j in the chromosome i, and $\bar{S}_i = \frac{\sum_{j=1}^{L} S_{ij}}{L}$, $\bar{S} = \frac{\sum_{i=1}^{L}\sum_{j=1}^{N} S_{ij}}{LN}$, and $\bar{S}_j = \frac{\sum_{i=1}^{N} S_{ij}}{N}$.
It is to be noted that the VAC and AVA will have low values when all the chromosomes in a population are almost identical. They used two FLCs for controlling p_c and p_m, depending on VAC and AVA, respectively. The fuzzy rules suggested were the following:
If VAC is low **then** p_c should be adjusted upwards slightly.
If VAC is high **then** p_c should be forced downwards slightly.
If AVA is low **then** p_m should be adjusted upwards slightly.
If AVA is high **then** p_m should be forced downwards slightly.

2.4.2 Approach 2 (Application of GA to improve the performance of an FLC):

The performance of an FLC depends on its knowledge base (KB) which consists of data base (DB) (that is, information regarding membership function distributions) and rule base (RB). It is important to mention that determination of an appropriate knowledge base for an FLC is not an easy task. The genetic algorithms (GAs) have been used by several investigators to design data base and/or rule base of an FLC. The fuzzy systems making use of a GA in their design process are called *genetic fuzzy systems* (GFS). There are, in fact, three different approaches of designing GFS, according to the KB components included in the learning process. These are as follows:

- Genetic Learning/Tuning of the Fuzzy Logic Controller Data Base.
- Genetic Learning/Tuning of the Fuzzy Logic Controller Rule Base.
- Genetic Learning/Tuning of the Fuzzy Logic Controller Knowledge Base.

Each of the above mentioned approaches is discussed below, as follows:

- **Genetic Learning/Tuning of the Fuzzy Logic Controller Data Base.**
  Karr (1991a) used a GA-based learning process for determining the data base (DB) of an FLC keeping its rule base (previously defined) fixed. He used a simple GA with binary coding, proportional selection scheme, simple crossover, and random mutation. He considered only triangular-shaped membership function distributions, in his study, as shown in Fig. 2.6.

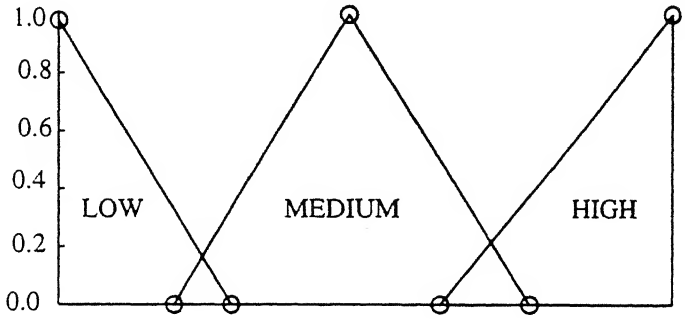


Figure 2.6: Membership function distributions (Karr 1991a).

The GA was used to move and to expand or shrink the base of each interior isosceles triangle. The extreme triangles will be right triangles and the GA will make it either bigger or smaller. In his study, the fitness function is expressed in terms of error.

Herrera et al. (1995a) proposed a GA-based tuning of a previously defined data base for an FLC. The process will start with a complete KB defined *a priori*. They used a real-coded GA with a stochastic universal sampling selection scheme, Michalewicz's non-uniform mutation operator, and a Max-Min-Arithmetical crossover. They assumed the membership function distributions to be trapezoidal. In their coding, each individual of the population represents a complete KB. Two individuals in the population are different only in their DB definition but their RB definition will be same. Thus, through evolution, the GA will find a good DB for the FLC.

- **Genetic Learning/Tuning of the Fuzzy Logic Controller Rule Base.**
  All methods belonging to this family assume the existence of a pre-defined DB for the FLC. The RB is represented in the form of a decision table (or, look-up table).

  Thrift(1991) suggested a method in which all the cells of the decision table are encoded in the chromosome. The term set associated to the output variable, (NB,

NS, ZR, PS, PB) is mapped into the set (0, 1, 2, 3, 4, 5) in which NB is represented by 0, NS by 1, ZR by 2, PS by 3, PB by 4 and the blank symbol – (absence of the control rule) by 5. He used an integer-coded GA with an elitist selection scheme, two-point crossover, and a specifically designed mutation operator. The mutation operator is such that when it is applied over an allele different from the blank symbol, changes it either up or down one level or to the blank code. Moreover, when the previous generation value is the blank symbol, it selects a new value at random.

Karr (1991b) implemented one method for deriving RB for an FLC in which the complete RB in the form of a decision table is not coded in each chromosome. His method is applicable only when the user is having deep knowledge about the system to be controlled. Moreover, the user should know the number of rules needed to control the system and some of the fuzzy rules *a priori*. Therefore, a chromosome codes only a little part of the complete RB. He used a simple GA with binary coding, proportional selection scheme, simple crossover and random mutation. In his approach, seven different linguistic terms describing the control variable are represented as a three-bit string (000 represents action 1 (NB), 001 represents action 2 (NM), and so on). Moreover, he used an application specific measure (that is, error) to define the fitness function.

Bonarini (1993) proposed a new technique known as *Evolutionary Learning of Fuzzy Rules (ELF)* for RB derivation of an FLC. He used a GA-based learning technique for rule base derivation (keeping its data base fixed) in which the whole decision table is coded for each individual in the population. Thus, the GA maintains a population of rules. To have a measure of fitness, each individual of the population, that is, each rule, will have associated information about several questions: how good it has been judged (its strength), when it has been generated, when it has been triggered the last time and how much it contributed to past control actions performed by the controller. It is to be noted that his approach is computationally expensive.

- **Genetic Learning/Tuning of the Fuzzy Logic Controller Knowledge Base.** There exists, in fact, many approaches for genetic learning of complete FLC KB. Some of these approaches are discussed below.

  Lee and Takagi (1993b) suggested one method for derivation of complete FLC KB using a GA. They used triangular-shaped membership functions, in their study,

although their method can work with any kind of parameterized membership functions, namely Gaussian, bell, trapezoidal or sigmoidal. They used a binary-coded GA. First, the information regarding shape of membership function distributions is coded in chromosomes of 24 bits long. The last part of the chromosome is built by coding the parameters w_i associated to each combination of the input values and joining them into a new binary substring. Eight bits are used again to encode the values of these parameters. In this way, each chromosome represents a complete KB. The fitness function is based on optimizing two different criteria, namely a measure of convergence and the number of rules present in the obtained RB.

Cooper and Vidal (1993) developed a novel approach for genetic learning of FLC KB in which they used a GA (with integer coding) with variable chromosomal length. A GA with an excessive length of the chromosome encoding the KB may not be able to find accurate solutions due to the high complexity of the search space. They proposed an encoding scheme which maintains only those rules necessary to control the systems. Thus, a GA will find the RB with the optimal number of rules. They considered triangular membership functions, specified by the location of its center and the half-length of its base. The fitness function is designed by using a measure of convergence.

Ng and Lee (1994) proposed a method for GA-based learning of complete FLC KB suitable for two-inputs-one-output systems, whose input and output spaces are partitioned into exactly seven primary fuzzy sets. It is important to note that their approach is suitable for a particular class of problems. The whole decision table (representing RB) is encoded as the part of chromosome. They used exponential membership function distributions, in their study, which are encoded in the second part of the chromosome. Thus, each chromosome represents a complete KB. In their approach, the fitness value is expressed as a measure of convergence. It is important to mention that their approach is not able to learn the number of rules that will constitute the optimal RB.

Liska and Melsheimer (1994) used a GA for simultaneously discovering fuzzy rules and membership functions, with a final stage of fine-tuning membership functions using the conjugate gradient descent method. They applied the system to learning a dynamic model of plant using known input-output data.

Eiji Nawa et al. (1997) proposed a pseudo-bacterial genetic algorithm (PBGA) with adaptive operator in the design of a fuzzy logic controller for a semi-active suspension

system. They introduced an adaptive operator in the PBGA to determine the points for the bacterial mutation and also the cutting points for the crossover operator. In their approach, each chromosome in the population encodes the rules of the rule base of the fuzzy model as well as the membership functions of the variables. Their approach was able to design an FLC with better quality rules.

Moreover, the work of Satyadas and Krishnakumar (1994), Carse and Fogarty (1994), Herrera et al. (1995b) are significant in this field of research.

2.5 Summary

An introduction is given to Soft Computing which includes fuzzy logic (FL) technique, neural networks (NNs), genetic algorithms (GAs), and their combinations. Soft Computing is used, now-a-days, to solve the complex real-world problems. In this chapter, the working principles of a fuzzy logic controller (FLC) and genetic algorithms (both binary-coded GA as well as real-coded GA) have been discussed, as these two techniques are used in the present work. Moreover, this chapter provides a literature survey on the combined GA-Fuzzy approaches developed by several researchers, in the past.

Chapter 3

NAVIGATION AMONG STATIC OBSTACLES

In this chapter, the working principle of a fuzzy-genetic algorithm has been explained and its effectiveness is tested on a number of find-path problems of a mobile robot. Moreover, the results of the proposed fuzzy-genetic algorithm have been compared to those of the two techniques, namely steepest descent method with a penalty function approach and tangent graph approach along with A* algorithm.

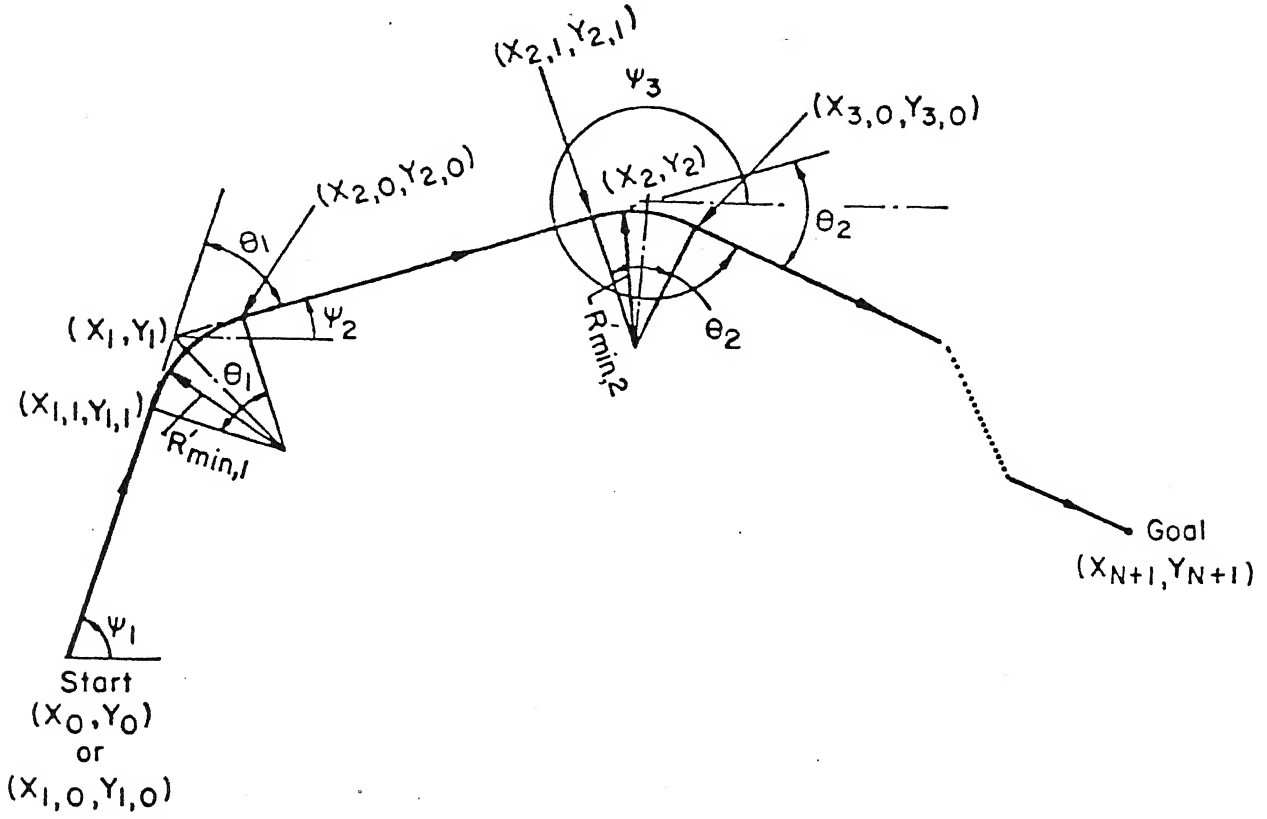
3.1 Mathematical Formulation of the Problem

The problem may be stated as follows: A mobile robot has to move from its initial position to a fixed final position by avoiding a set of fixed obstacles of different sizes. The objective is to find a path that takes minimum time of travel and also avoids hitting the obstacles.

The following assumptions are made to simplify the problem:

1. Robot is considered to be a single point.
2. Each obstacle is represented by its bounding circle.
3. The obstacles are disjoint, that is, no two obstacles are allowed to overlap at any time.

With these assumptions, the physical motion planning problem is converted into a geometrical path planning problem. This problem in robotics is widely known as *find-path* problem. The geometric path from the starting point to the goal is assumed to be a sequence of straight-line segments with intermediate circular arcs, as shown in Fig. 3.1. Thus, the trajectory is fully designated by a number of control points (X_n, Y_n). The



Ψ_p : Angle of p-th straight line segment with the horizontal axis.

θ_n : Included angle between two consecutive straight-line segments.

Figure 3.1: Proposed trajectory as a sequence of straight-line segments.

velocity and acceleration distributions along the trajectory are shown in Fig. 3.2. The

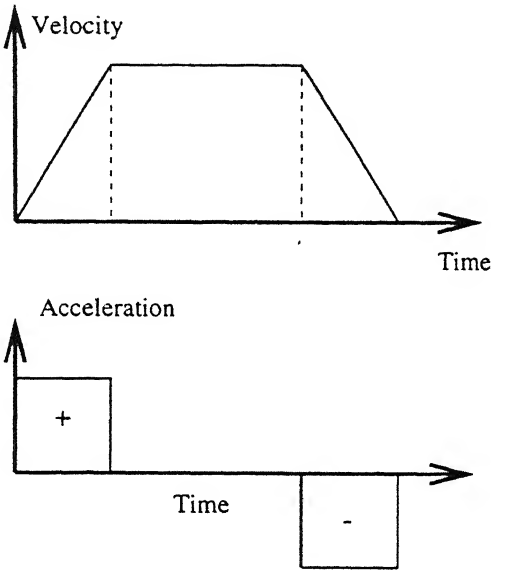


Figure 3.2: Velocity and acceleration distributions along the trajectory.

robot starts accelerating from the initial position and reaches a maximum velocity v. It then moves with this maximum velocity through some distance and then decelerates so as to reach the final point with zero velocity.

The find-path problem is treated as the traveling time minimization problem subject to satisfying the constraints due to the presence of obstacles, velocity limits, and acceleration limits. The problem is stated mathematically as follows:

$$\begin{array}{ll} \text{Minimize} & \text{Traveling time, } T = \frac{D}{v} + \frac{v}{a} \\ \text{subject to} & \text{the path is obstacle-free and} \\ & X_{\min} \leq X_n \leq X_{\max} \quad n = 1, 2, \ldots, N \\ & Y_{\min} \leq Y_n \leq Y_{\max} \quad n = 1, 2, \ldots, N \\ & v_{\min} \leq v \leq v_{\max} \\ & a_{\min} \leq a \leq a_{\max} \end{array} \tag{3.1}$$

where D is the total distance traveled along the trajectory, v is the maximum velocity of the robot along the trajectory, a is the acceleration, and N indicates the total number of control points in the resultant trajectory.

The first term in the objective function (Equation 3.1) is the average time taken for traveling from an initial point to a final point and the second term indicates the time of

initial acceleration and final deceleration during the travel (Fig. 3.2). The total distance D can be computed as the sum of straight-line distances and curved distances. In order to achieve the exact straight path, two points $(X_{n,1}, Y_{n,1})$ and $(X_{n+1,0}, Y_{n+1,0})$ are derived on either sides of each control point (X_n, Y_n), as shown in Fig. 3.1. Thereafter, the total distance is computed as follows:

$$D = \sum_{n=1}^{N+1} \sqrt{(X_{n,1} - X_{n,0})^2 + (Y_{n,1} - Y_{n,0})^2} + \sum_{n=1}^{N} \acute{R}_{\min,n}\, \theta_n, \tag{3.2}$$

where

$$X_{1,0} = X_0,\ Y_{1,0} = Y_0 \text{ and } X_{N+1,1} = X_{N+1},\ Y_{N+1,1} = Y_{N+1}.$$

(X_0, Y_0) and (X_{N+1}, Y_{N+1}) are the coordinates of the initial and final positions, respectively. The minimum radius of curvature corresponding to the n-th intermediate control point is computed as follows:

$$\acute{R}_{\min,n} = \frac{v^2}{a}. \tag{3.3}$$

The parameter θ_n is the included angle between two consecutive straight-line segments at the n-th control point and it is calculated as follows:

$$\theta_n = 2\tan^{-1} \frac{\sqrt{(X_{n+1,0} - X_n)^2 + (Y_{n+1,0} - Y_n)^2}}{\acute{R}_{\min,n}}. \tag{3.4}$$

In order to ensure an obstacle-free trajectory, a number of constraints are introduced. For simplicity, each of the M obstacles (whether convex or concave) is approximated by an enclosing circle. Then, all obstacles are assumed to be circular having centers at (X_c^m, Y_c^m) and having radius r^m. To make the matter simple, all control points (excluding the initial and final points) are checked to investigate whether the points are lying on the obstacles. Thus, a constraint for each control point is used as follows:

$$\sqrt{(X_n - X_c^m)^2 + (Y_n - Y_c^m)^2} \geq r^m \quad m = 1, 2, \ldots, M. \tag{3.5}$$

This constraint is handled using the following penalty function

$$P_1 = \begin{cases} \sum_{n=1}^{N} \sum_{m=1}^{M} 1/\left[(X_n - X_c^m)^2 + (Y_n - Y_c^m)^2\right], & \text{if } \sqrt{(X_n - X_c^m)^2 + (Y_n - Y_c^m)^2} < r^m \\ 0, & \text{otherwise.} \end{cases} \tag{3.6}$$

This requires at most $3NM$ floating point evaluations (flops). Ensuring all control points to be free of obstacles does not guarantee that the whole trajectory as defined earlier

is obstacle-free. In addition to the above constraints, all straight-line segments are also checked for obstacle avoidance. For each straight-line segment, the perpendicular distance d_n^m from the center (X_c^m, Y_c^m) of each obstacle-circle is computed and the constraint can be expressed as follows:

$$d_n^m \geq r^m \quad n = 1, 2, \ldots N; \quad m = 1, 2, \ldots M. \tag{3.7}$$

The following penalty factor is used to ensure an obstacle-free trajectory

$$P_2 = \begin{cases} \sum_{n=1}^{N+1} \sum_{m=1}^{M} 1/(d_n^m)^4, & \text{if } d_n^m < r^m \\ 0, & \text{otherwise.} \end{cases} \tag{3.8}$$

For N control points and M obstacles, there are a total of $(N+1)M$ such computations. The shortest point on the n-th line-segment from m-th circle is calculated as follows:

$$\begin{pmatrix} \alpha \\ \zeta \end{pmatrix} = \begin{pmatrix} X_{n,0} \\ Y_{n,0} \end{pmatrix} + t^* \begin{pmatrix} X_{n,1} - X_{n,0} \\ Y_{n,1} - Y_{n,0} \end{pmatrix}, \tag{3.9}$$

where

$$t^* = \frac{(X_c^m - X_{n,0}) + \nu(Y_c^m - Y_{n,0})}{(X_{n,1} - X_{n,0}) + \nu(Y_{n,1} - Y_{n,0})}, \quad \text{and}$$
$$\nu = \frac{(Y_{n,1} - Y_{n,0})}{(X_{n,1} - X_{n,0})}.$$

This requires 4 flops to calculate t^*. If t^* lies within $(0, 1)$, then the squared shortest distance is calculated as follows:

$$(d_n^m)^2 = (X_c^m - \alpha)^2 + (Y_c^m - \zeta)^2. \tag{3.10}$$

It requires 4 more flops. It is noteworthy that this computation of squared distance need not be calculated for all n and m combinations. However, in the worst case, there are a total of $10(N+1)M$ flops required to calculate P_2. Thus, each evaluation of a trajectory for obstacle avoidance has linear computational time complexity with either the number of obstacles (M) or the number of control points (N). It is to be noted that no penalty term is considered for taking care of the circular arcs to be obstacle-free, in this study.

3.2 Proposed Algorithm

The proposed algorithm is based on GA-Fuzzy combination (refer to section 2.4). The main drawback of a GA is its slow convergence rate and chance of premature convergence.

Here, a fuzzy logic technique is used to improve the performance of a GA (refer to section 2.4.1). The proposed algorithm (that is, *fuzzy-genetic algorithm*) uses fuzzy rule base to develop efficient GA operators (crossover and mutation) and to create initial population for GA. Fig. 3.3 shows the schematic diagram of a fuzzy-genetic algorithm. Since the

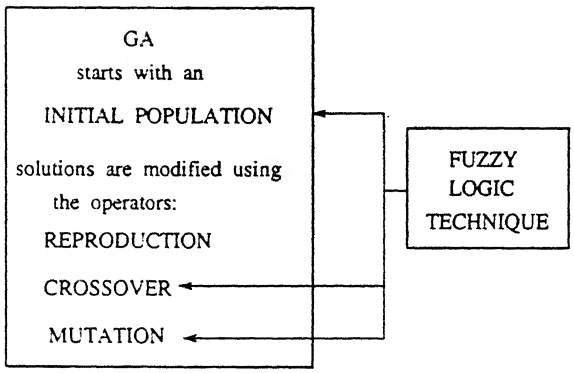


Figure 3.3: A schematic diagram showing fuzzy-genetic algorithm.

variables (coordinates of control points, velocity and acceleration) are real-valued, a real-coded GA, as discussed in section 2.3.2, is used in this study. A penalty function method has been used to drive the search towards the collision-free paths in the presence of stationary obstacles. In the following, the fuzzy-genetic algorithm is described in detail.

3.2.1 Evaluation of a Solution

From the initial point, final point and the location of the obstacles, a search direction s_1 (refer to Fig. 3.6) from the initial point is computed using a fuzzy logic technique. The inputs of the fuzzy logic controller (section 2.2.2) are the distance from the initial point to the center of the nearest obstacle in the direction of final point and the angle between the line joining the initial and final points and the line joining the initial point and the center of nearest obstacle. The output of the fuzzy logic controller is the direction s_1 (in terms of deviation of the first line segment from the initial point). Fig. 3.4 shows a schematic diagram indicating condition (distance and angle) and action (deviation) variables of the FLC. Fig. 3.5 shows the membership function distributions used for distance, angle and deviation (same as angle). For simplicity, the membership function distributions are assumed to be triangular. Since there are five options in the distance metric and eight options in the angle metric, there are a total of 5×8 or 40 different rules possible. Table

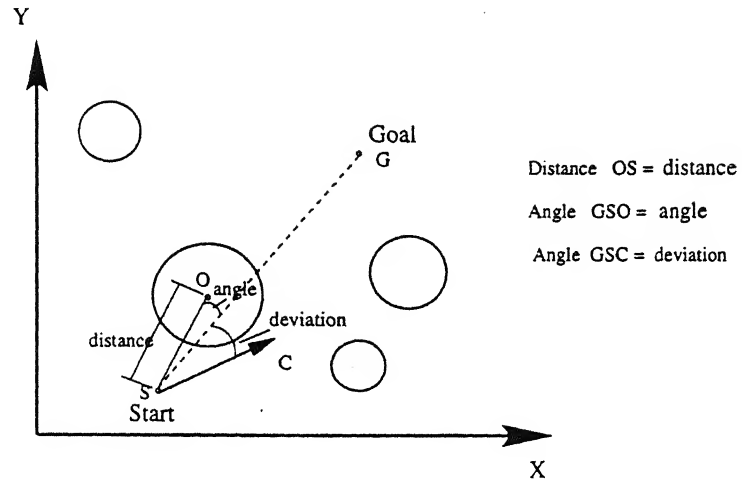


Figure 3.4: A schematic diagram showing condition (distance and angle) and action (deviation) variables of the FLC.

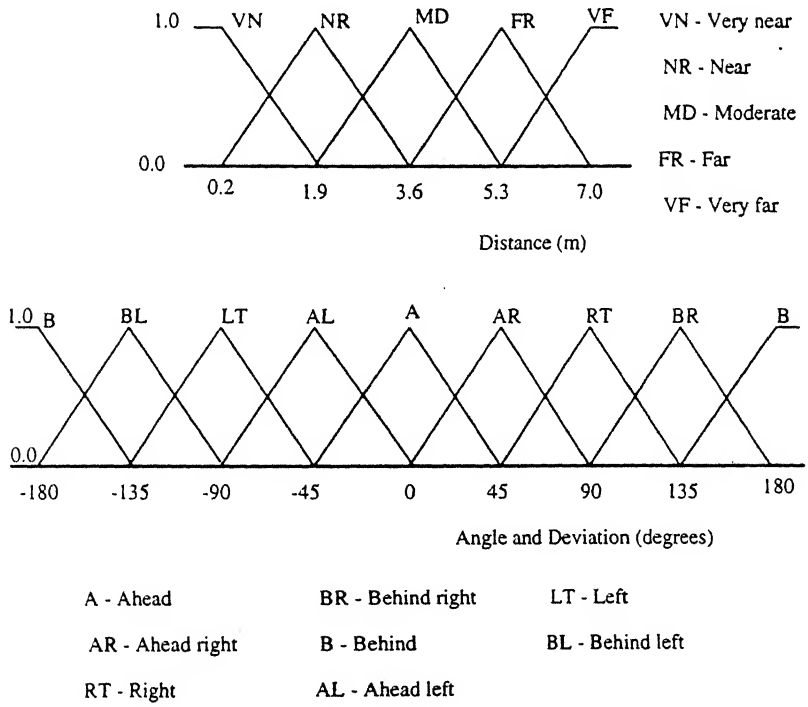


Figure 3.5: Membership function distributions for distance, angle and deviation.

3.1 shows the author-defined rule base (consisting of 40 rules) of the FLC.

Table 3.1: Author-defined rule base of an FLC (find-path problem)

distance \ angle	A	AL	LT	BL	B	BR	RT	AR
VN	RT	AR	A	A	A	A	A	AL
NR	AL	A	A	A	A	A	A	AL
MD	AR	A	A	A	A	A	A	A
FR	AR	A	A	A	A	A	A	A
VF	A	A	A	A	A	A	A	A

For example, three different typical rules used in the rule base are shown here:

If distance is VN and angle is AL, then deviation is AR
If distance is VN and angle is LT, then deviation is A
If distance is NR and angle is AR, then deviation is AL

The following steps are used to find an obstacle-free direction for the robot:

Step 1: Calculate the distance and angle of the nearest obstacle from the robot.

Step 2: From the membership functions, find which two options of distance and angle are affected. For example, if distance is 2 m and angle is 20°, Fig. 3.5 shows that membership functions for distances *NR* (near) and *MD* (moderate) include the distance of 2 m. Similarly, angles *A* (ahead) and *AR* (ahead-right) are invoked.

Step 3: With two options of distance and angle, check which four of 40 rules are fired. The output of the fuzzy logic controller is determined using the method as explained in section 2.2.2. The De-fuzzification gives a deviation, which is a local obstacle-free direction.

It is noteworthy that, for a particular configuration of obstacles and initial and final points, the first direction, s_1 (refer to Fig. 3.6), obtained using the above principle is a fixed direction for all solutions created during the optimization process. It is now

discussed how subsequent directions (s_2, s_3, and so on) and the complete trajectory is determined from a solution represented in a GA as follows:

$$(d_1, d_2, \ldots d_N, v, a).$$

The variables d_1, d_2, ..., d_N are distances along the directions s_1, s_2, and so on. The variables v and a are the velocity and acceleration of the robot along the path, respectively.

Once the first line segment direction s_1 is known, the first control point (X_1, Y_1) is found along s_1 at a distance d_1 from the initial point (X_0, Y_0). In order to locate the second control point, again an obstacle-free direction s_2 is found at the first control point using the fuzzy rule base with an appropriate input distance and angle, and then by traveling along s_2 a distance d_2 from the first control point (X_1, Y_1). This procedure is repeated N times to locate N control points and finally the path is directed towards the final point from the N-th control point. Here, an obstacle-free direction is determined locally using an FLC and the extent of travel along the obstacle-free direction is found using a GA. To achieve a smooth transition from one line segment to another, an intermediate circular arc is added as shown in Fig. 3.1.

Once a trajectory is found, the overall objective function is computed as follows:

$$F = T + P_1 + P_2. \tag{3.11}$$

3.2.2 Reproduction Operator

A tournament selection scheme (Goldberg and Deb 1991) is used here, in which two solutions are compared in a pair and the best (in terms of objective function value) is selected.

3.2.3 Crossover Operator

Real-coded GAs deal with real variables and an efficient crossover operator known as simulated binary crossover (SBX) (Deb and Agrawal 1995) is used here. Consider the two parent solutions:

$$\text{Parent 1:} \quad d_1^1, d_2^1, \ldots, d_N^1, v^1, a^1$$

$$\text{Parent 2:} \quad d_1^2, d_2^2, \ldots, d_N^2, v^2, a^2$$

A crossover probability p_c is used to check whether they have to be crossed. If so, the crossover operator as described in Section 2.3.2 is used, otherwise they are kept unchanged.

Using crossover, two new solutions are computed as

$$\text{Child 1:} \quad D_1^1, D_2^1, \ldots, D_N^1, V^1, A^1$$

$$\text{Child 2:} \quad D_1^2, D_2^2, \ldots, D_N^2, V^2, A^2$$

Note that on an average 50% of these values are the same as those in the parent solution and other 50% are new values near the parent values created using the SBX operator. Once these children solutions are found, each of them is evaluated as above using the fuzzy rule base to form a complete trajectory. The procedure is illustrated in Fig. 3.6. The figure shows two children points (a_1 and a_2) in the first line segment obtained by

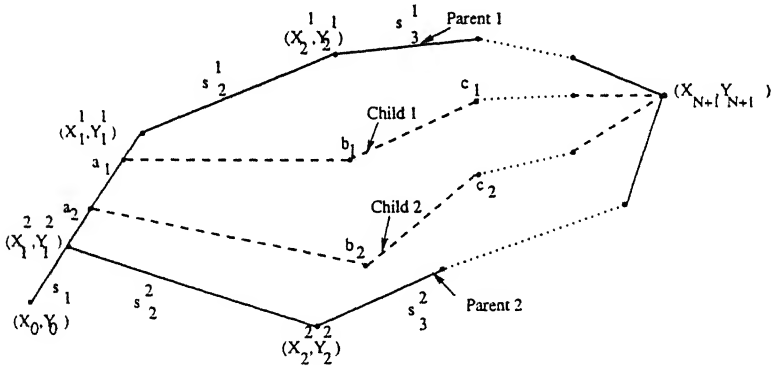


Figure 3.6: Crossover operation.

performing SBX on the points (X_1^1, Y_1^1) and (X_1^2, Y_1^2). Similarly, the second control points (b_1 and b_2) and other control points of the children solutions are obtained.

Using the fuzzy rule base to create two new solutions has an advantage in the GA search. The performance of GA largely depends on the *building block processing* inherent to the coding and the operators used (Goldberg 1989, Kargupta et al. 1992). Since fuzzy rule base always creates an obstacle-free direction at least in the neighborhood of a control point and SBX creates near-parent solutions with a large probability, the resulting solutions after crossover are expected to be good.

3.2.4 Mutation Operator

Each component of a solution vector is checked for mutation with a mutation probability p_m. If a value is to be mutated, the procedure as described in Section 2.3.2 is used. Fig. 3.7 shows both the original path as well as the mutated path. Starting from the

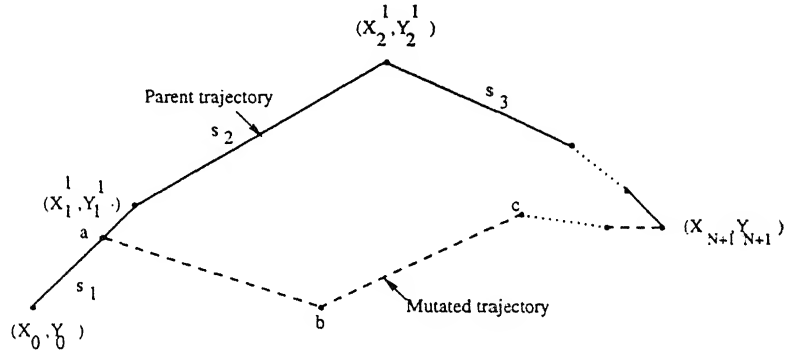


Figure 3.7: Mutation operation.

initial position up to the point a, directions of both current and mutated paths are the same, as indicated by s_1. At point a, the mutated path finds a new direction (different from s_2) using the fuzzy rule base (refer to Section 2.2.2). The other intermediate points ($b, c, \ldots$ etc) of the mutated path are computed following the same method, as described in Section 3.2.1.

3.3 Simulation Results and Discussion

The effectiveness of the proposed algorithm is tested on a number of find-path problems of a mobile robot (Pratihar et al. 1998b, 1999c). The robot starts with a zero velocity and its maximum velocity and acceleration are set to 2 m/sec and 2 m/sec^2, respectively, in this study. Four different cases are considered. In the first and second cases, five ($M = 5$) and six ($M = 6$) obstacles are considered, respectively and only two control points ($N = 2$) are used to find collision-free path. Similarly, eight ($M = 8$) and ten ($M = 10$) obstacles are assumed in the third and fourth cases, respectively and three control points ($N = 3$) are considered. All these cases are discussed below, in detail.

Case 1

Fig. 3.8 shows the find-path problem of a mobile robot in the presence of five static obstacles (in a grid size of 14×14 m^2). The point robot will have to find a time-optimal and collision-free path while moving from an initial position S to the final position G. In order to illustrate the complexity of this apparently simple find-path problem, an attempt is made to solve it with a *steepest descent method* (Deb 1995) using the penalty function approach. It is observed that the resulting path largely depends on the chosen initial solution. Table 3.2 shows the initial solution and the corresponding final solution for four sub-cases obtained using the steepest descent method.

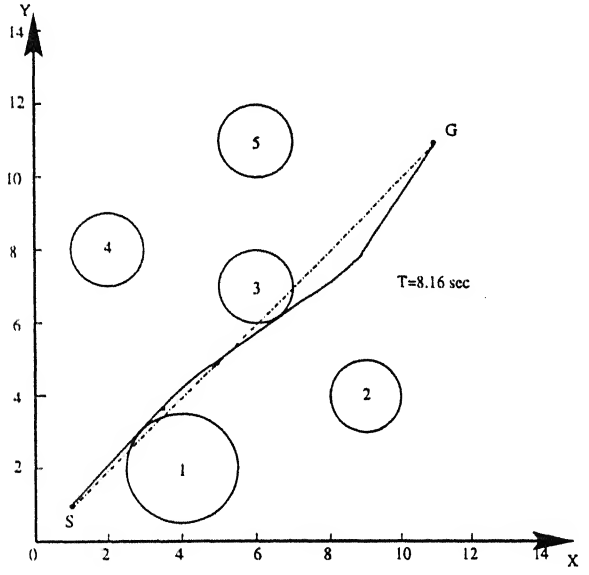


Figure 3.8: Optimized path obtained using fuzzy-genetic algorithm in 5-obstacles case.

Table 3.2: Results of the steepest descent method (Case 1)

Initial solution $(X_1, Y_1, X_2, Y_2, v, a)$	Nature of solution	Final solution $(X_1, Y_1, X_2, Y_2, v, a)$	Nature of solution	Traveling time, T,sec
(4.0, 7.0, 6.0, 9.0, 1.0, 1.2)	obstacle-free	(1.0, 11.0, 4.5, 9.6, 2.0, 2.0)	obstacle-free	13.49
(6.0, 3.0, 6.0, 9.0, 1.0, 1.2)	Not obstacle-free	No Solution		
(4.0, 7.0, 9.0, 7.0, 1.0, 1.2)	Not obstacle-free	No Solution		
(4.0, 6.0, 7.0, 8.0, 1.0, 1.2)	Not obstacle-free	(3.2, 6.2, 9.2, 5.7, 2.0, 1.7)	Obstacle-free	10.71

It is clear from the table that in most of the cases, the obtained solutions are neither feasible nor collision-free. Further, the dependence of the final solution on the initial solution does not make this approach attractive. Thus, the steepest descent method is not used in the subsequent cases, instead the results are presented only with the proposed approach.

Fig. 3.8 shows the locations of the stationary obstacles, the initial and final positions of the robot. The following GA parameters are used after a careful study:

Number of generations = 40
Crossover probability = 0.9
Mutation probability = 0.005

A population size of 40 is used here for a GA run. The minimum traveling time has been obtained as 8.16 sec and the collision-free near time-optimal path is shown in Fig. 3.8. In order to investigate the efficacy of the proposed algorithm, the optimized path is found using the combined *tangent-graph and A* method* (Liu and Arimoto 1991, 1992, 1995) also. In this case, the path obtained is very similar to that in Fig. 3.8. The path begins from the starting point, S and then goes tangentially to the obstacle 1. Thereafter, the path is a common tangent to obstacles 1 and 3 and finally after traversing along the boundaries of obstacle 3, the path goes out tangentially towards the goal. The overall traveling time is 8.08 sec, which is 0.08 sec smaller than that found using the proposed algorithm. It will be shown later that the complexity of the proposed method is linear to the number of control points, whereas the complexity of the tangent-graph and A* method is quadratic to the number of convex segments of obstacle boundaries. A few more difficult problems have been solved here.

Case 2

The proposed algorithm is tested on a more complicated find-path problem (in a grid size of 12×12 m^2), shown in Fig. 3.9. It is important to note that, till now, no concrete convergence proof (mathematically) of a GA is available in the literature. Thus, a question may arise - whether GA can be used as an optimizer, at all. To answer this, an experiment has been carried out, as described below, on sub-case 1 of Fig. 3.9 (that is, initial point

S_1 and final point G_1). As the performance of a GA depends on its different parameters, one parameter is varied, at a time, keeping the others fixed and the obtained values of traveling time are noted down in the tabular form.

Table 3.3: Variations in traveling time by varying population size and keeping number of generations, p_c, p_m fixed

Sl. No.	Population size	Traveling time, sec
1	10	8.25
2	20	8.00
3	30	7.94
4	40	7.94
5	50	7.93
6	60	7.93
7	70	7.93

Table 3.4: Variations in traveling time by varying number of generations and keeping population size, p_c, p_m fixed

Sl. No.	Number of generations	Traveling time, sec
1	10	8.38
2	20	8.35
3	30	8.25
4	40	7.94
5	50	7.94
6	60	7.93
7	70	7.93

Table 3.5: Variations in traveling time by varying p_c and keeping population size, number of generations, p_m fixed

Sl. No.	p_c	Traveling time, sec
1	0.8	8.20
2	0.85	7.93
3	0.9	7.93
4	0.92	7.97
5	0.95	7.96

Table 3.6: Variations in traveling time by varying p_m and keeping population size, number of generations, p_c fixed

Sl. No.	p_m	Traveling time, sec
1	0.001	7.95
2	0.005	7.46
3	0.01	7.93
4	0.02	7.98
5	0.03	8.03

Table 3.3 shows the values of traveling time obtained by varying the population size and keeping the number of generations, crossover probability (p_c), mutation probability (p_m) fixed. Similarly, the variation in traveling time is shown in Table 3.4 obtained while varying the number of generations (in the range of 10 to 70) and keeping the population size, p_c, p_m fixed. Moreover, the p_c has been varied from 0.8 to 0.95 and the values of traveling time are noted down in a tabular form (refer to Table 3.5) keeping the population size, number of generations and p_m fixed to be 60, 60, 0.01, respectively. Table 3.6 shows the different values of traveling time obtained by varying p_m alone (in the range of 0.001 to 0.03) and keeping the population size, number of generations, p_c fixed to the values 60, 60, 0.9, respectively.

From the above study, it may be concluded that a GA can improve the objective function value and the best performance of a GA is seen with a particular set of parameters. Here, the following GA parameters are used:

Number of generations = 60
Crossover probability = 0.9
Mutation probability = 0.01

The population size is set to 60. The minimum traveling time for two different sub-cases, that is, starting from two different initial positions (S_1 and S_2) to the same final position (G_1) are shown in Table 3.7.

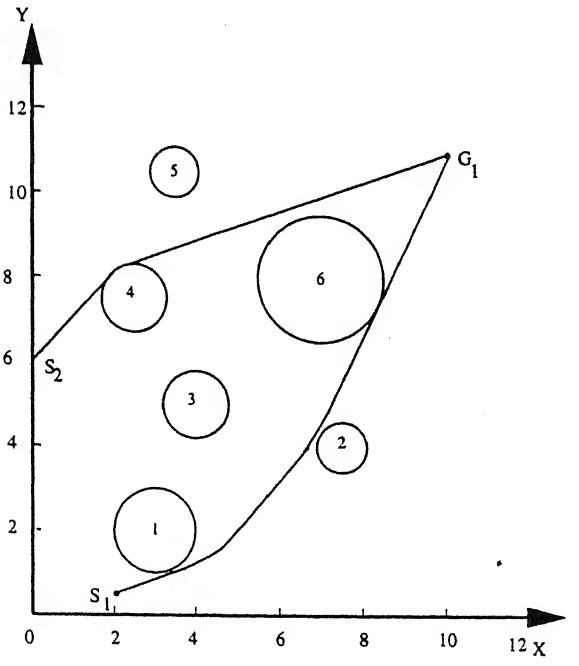


Figure 3.9: Optimized path obtained using fuzzy-genetic algorithm in 6-obstacles case.

Table 3.7:Results of the proposed algorithm and the tangent-graph technique (Case 2)

Sl. No.	Path		Traveling time, T (sec)	
	Start	Goal	Proposed	Tangent-graph & A*
Sub-case 1	S_1	G_1	7.93	7.86
Sub-case 2	S_2	G_1	6.72	6.65

Fig. 3.9 shows the near time-optimal collision-free path for each of these sub-cases. Table 3.7 also shows the corresponding traveling time of the optimized path found using the combined tangent-graph and A* algorithm. The table shows that in all cases, the traveling time along the path obtained using the proposed fuzzy-GA approach is close to that obtained using tangent-graph and A* approach.

Case 3

Fig. 3.10 shows the scenarios in which the robot will have to find time-optimal collision-free path while moving among eight static obstacles (in a grid size of 14×14 m^2). The following GA parameters are selected after a careful study:

Number of generations = 60
Crossover probability = 0.92
Mutation probability = 0.01

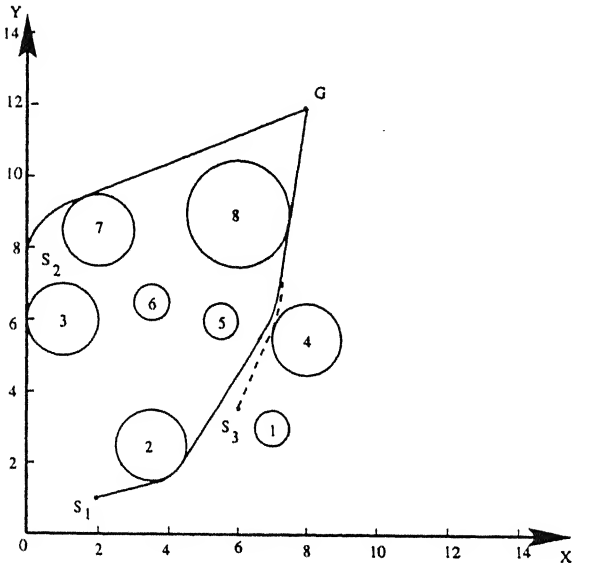


Figure 3.10: Optimized path obtained using fuzzy-genetic algorithm in 8-obstacles case.

A population size of 80 is used in the GA run. The minimum traveling time for three different sub-cases (that is, starting from three different initial positions to the same final position), as obtained by the proposed algorithm and the tangent graph technique are shown in Table 3.8. The traveling time along the path obtained using the proposed algorithm is found to be either close to or same as that obtained using the tangent-graph and A* approach. Fig. 3.10 shows the near time-optimal collision-free path for each of these sub-cases.

Table 3.8: Results of the proposed algorithm (Case 3)

Sl. No.	Path		Traveling time, T (sec)	
	Start	Goal	Proposed	Tangent-graph & A*
Sub-case 1	S_1	G	7.84	7.50
Sub-case 2	S_2	G	5.61	5.52
Sub-case 3	S_3	G	5.39	5.39

Case 4

Fig. 3.11 shows two more difficult find-path problems defined in the grid size of 12×13 m^2. In the first problem, S_3 and G_2 are the initial point and the final point, respectively, whereas in the second problem, starting point is S_4 and goal is G_2. In all GA runs, crossover probability (p_c), mutation probability (p_m), maximum number of generations and population size are set to 0.94, 0.01, 60 and 80, respectively, after a careful study.

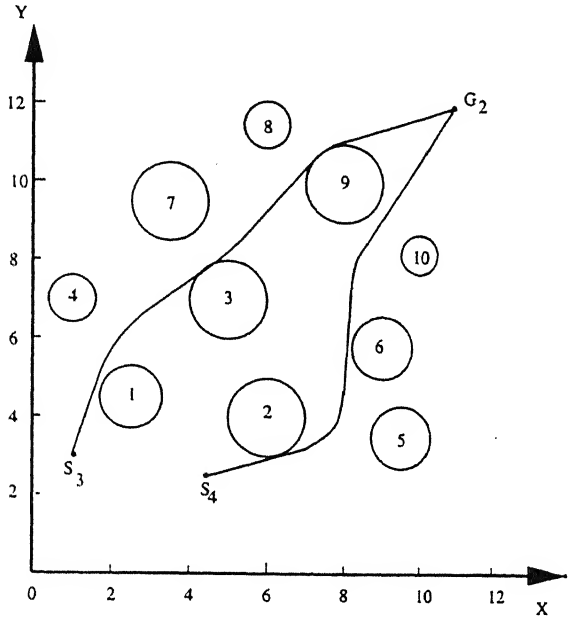


Figure 3.11: Optimized path obtained using fuzzy-genetic algorithm in 10-obstacles case.

These two problems are solved using both the proposed algorithm as well as the tangent-graph technique along with A* algorithm. It is seen that the results obtained by the proposed fuzzy-genetic algorithm are close to those of the combined tangent-graph and A* algorithm (refer to Table 3.9). Fig. 3.11 shows the near time-optimal collision-free trajectory obtained by the proposed algorithm.

Table 3.9:Results of the proposed algorithm and tangent-graph technique (case 4)

Sl. No.	Path		Traveling time, T (sec)	
	Start	Goal	Proposed	Tangent-graph & A*
Sub-case 1	S_3	G_2	8.14	8.02
Sub-case 2	S_4	G_2	7.31	6.81

It is clear that the path obtained by the proposed method is intuitively what a human would also take to reach the destination. The difference in the optimal time between the proposed method and the best-known tangent-graph is little and is primarily because in the latter method a robot always traverses along obstacle boundaries and their common tangents. As the performance of an FLC is dependent on its knowledge base, the accuracy of the proposed algorithm can be further improved by proper tuning of the knowledge base. Moreover, the proposed algorithm has an advantage in execution time, as discussed in the next subsection.

3.3.1 Time Complexity Study

In order to investigate the effect of number of control points (N) on the solution accuracy and on the computational time, an experiment is carried out with different values of $N = 3, 4$ and 5 on sub-case 1 of the fourth case having ten obstacles. For the same

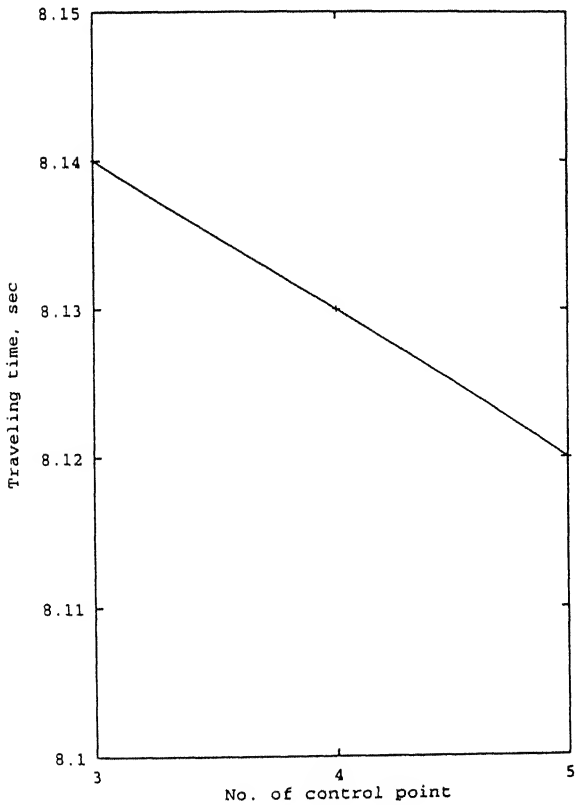


Figure 3.12: Traveling time versus number of control point.

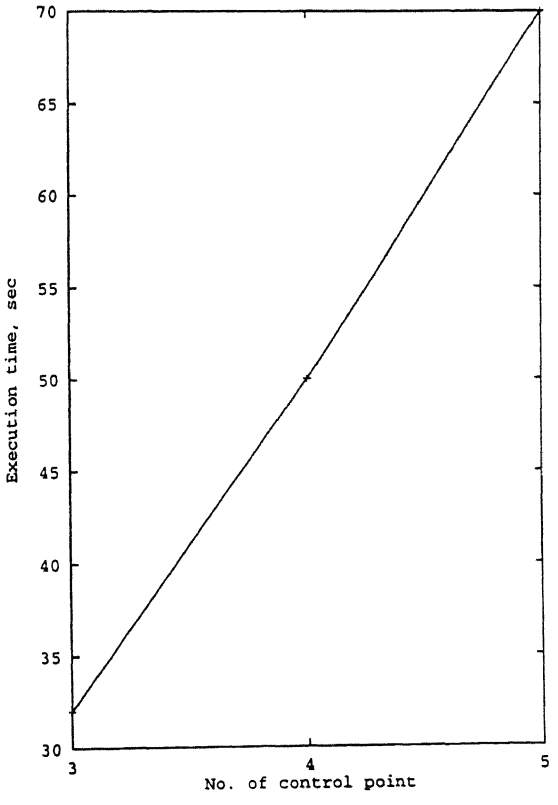


Figure 3.13: Execution time versus number of control point.

starting point S_3 and the goal G_2 (refer to Fig. 3.11), the traveling time as well as the execution time are recorded as obtained by the proposed fuzzy-genetic algorithm. All simulation runs are performed on a PC-386. The traveling time versus the number of control point and the execution time versus the number of control point are plotted in Figs. 3.12 and 3.13, respectively. These figures (Figs. 3.12 and 3.13) show that as the number of control point increases there is a slight improvement of the solution accuracy but the execution time increases linearly. This study shows that the complexity of the proposed algorithm is linear to the number of control points (the similar remarks can be made from the calculation of flops- section 3.1). On the other hand, the complexity of the tangent-graph and A* algorithm is quadratic (Liu and Arimoto 1991) to the number of control points and it becomes computationally intractable particularly for a reasonably large number of obstacles.

3.4 Summary

This chapter describes a fuzzy-genetic algorithm in which a fuzzy logic technique is used to improve the performance of a genetic algorithm. The time-minimal path planning problems of a mobile robot in the presence of static obstacles (known as find-path problems) are solved by using the proposed fuzzy-genetic algorithm. The proposed algorithm is found to perform better than a steepest descent method with a penalty function approach. More-over, the results obtained by the proposed fuzzy-genetic algorithm are found to be similar to those of the tangent graph technique along with A* algorithm, although the former is computationally faster than the latter.

Chapter 4

NAVIGATION AMONG MOVING OBSTACLES

This chapter explains the working principle of a genetic-fuzzy system (in which the performance of an FLC is improved by using a GA) and its effectiveness is tested on a number of navigation problems of a mobile robot in the presence of moving obstacles. Six different approaches are studied here and their performances have been compared.

4.1 Description of the Problem

The problem may be stated as follows: A mobile point robot (as assumed) has to move from its initial position to a fixed final position by avoiding a set of moving obstacles in minimum traveling time. The following assumptions are made to simplify the problem:

1. Each moving obstacle is represented approximately by its bounding circle.
2. The obstacles are disjoint, that is, no two obstacles are allowed to overlap at any time.
3. The motion of the robot is constrained by the moving obstacles only.

The navigation problems of a mobile robot in the presence of moving obstacles can be solved by using the *fuzzy-genetic algorithm*, as explained in the Section 3.2, but it is done

in two stages. At first, an optimal/near-optimal trajectory is generated considering the obstacles to be stationary and in the second stage, velocity of the robot is planned along this trajectory to avoid collisions with the moving obstacles. This method has its inherent limitations. The approach will fail if an obstacle is moving along the same path as the robot. Moreover, the sudden change of velocity will result into a jerky motion of the robot. This method is found to be not attractive for solving the navigation problems of a mobile robot among moving obstacles and a different approach based on GA-Fuzzy combination (known as *genetic-fuzzy approach*) is adopted here.

4.2 Proposed Genetic-Fuzzy Approach

There is a natural connection between the navigation problem of robot in the presence of moving obstacles and a combined approach of genetic algorithm and fuzzy logic technique. The purpose of the navigation problem of a robot is to find an obstacle-free path which takes a robot from a point S to a point G with minimum time. There are essentially two parts of the problem:

1. learn to find *any* path from point S to G that avoids all obstacles, and
2. learn to choose that obstacle-free path which takes the robot in a minimum possible time.

Both these problems are somewhat similar to the growing-up (learning) process of a child. If a child is kept in a similar (albeit hypothetical) situation (that is, a child has to go from one corner of a room to another corner by avoiding a few moving objects), one of the probable approaches the child may follow is take each object at a time. When an object is very near to the child, he may either stop to let the object pass by or he may take a small detour so that he avoids hitting the object. It is clear that when the child is taking a detour there is no particular angle by which he would turn when faced with a similar situation again. To avoid hitting an object, his objective is to deviate from his original path. This process of avoiding an object can be thought as if the child is using a rule of the following sort:

If an object is very near and is coming straight to him, then he turns right to his original path.

When such a situation happens, the most important thing for the child to do is to deviate from his path to avoid the imminent object. The exact angle of deviation is not that important. Thus, the angle of deviation can be imprecisely set, although an optimum angle of deviation can be computed using principles of physics and this computation will be extensive. Thus, it would make sense to use a fuzzy logic technique to find a suitable angle of deviation quickly than to use an exact angle calculated with unnecessary rigor.

The second task is to find an obstacle-free path which requires minimum possible time to reach from point S to G. This task is similar to the above-mentioned child simile, but relates to the way an inexperienced and an experienced child will solve the same problem. An inexperienced child may take avoidance of each obstacle too seriously and deviate by a large angle each time he faces an obstacle. This way, this child may lead away from the target point G and may finally reach G after traversing a long winding distance, whereas an experienced child may deviate barely from each obstacle, thereby taking the quickest route. It is important to mention that the experienced child has learnt this trick through experience of solving many such problems in the past. If it is assumed again that the child uses rules to do the task, the child has discovered (from experience) a set of efficient rules by solving similar tasks in the past. This is precisely an optimization procedure where an optimal set of rules are discovered which minimizes the travel time in the presence of moving objects. This learning process can be simulated by using an optimization algorithm—genetic algorithm (GA)—in training a robot to learn to find an optimal set of rules by simulating its motion in a number of user-defined scenarios.

Thus, the use of fuzzy logic technique helps in quickly determining imprecise yet obstacle-free paths and the use of a genetic algorithm helps in learning an optimal set of rules that a robot should use while navigating in the presence of moving obstacles. This process is illustrated in Fig. 4.1. A GA is used to create the optimal fuzzy knowledge base of a robot off-line. For on-line application, the robot uses its optimal fuzzy rule base to find an obstacle-free path for a given input parameters depicting the state of moving obstacles and the state of the robot. It is important to note that it would not be wise to solve both the above tasks: (i) finding a deviation to avoid an obstacle and (ii) finding a complete obstacle-free path which is shortest, independently. Both the problems are

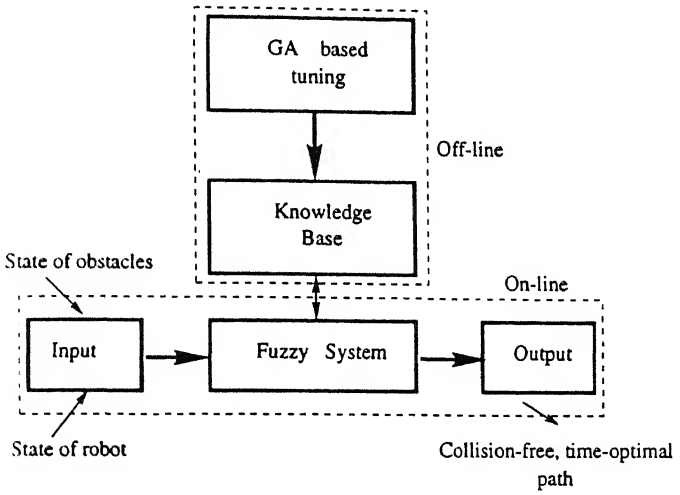


Figure 4.1: Genetic-fuzzy approach.

dependent on each other, in fact, the child (in the simile above) also does not solve both problems independently. The combined genetic-fuzzy approach has been devised to solve the navigation problem of a mobile robot among moving obstacles.

4.2.1 Representation of a Solution

A solution to the navigation problem is represented by a set of rules which a robot will use to navigate from point S to point G (Fig. 4.2). Each rule has three conditions: distance,

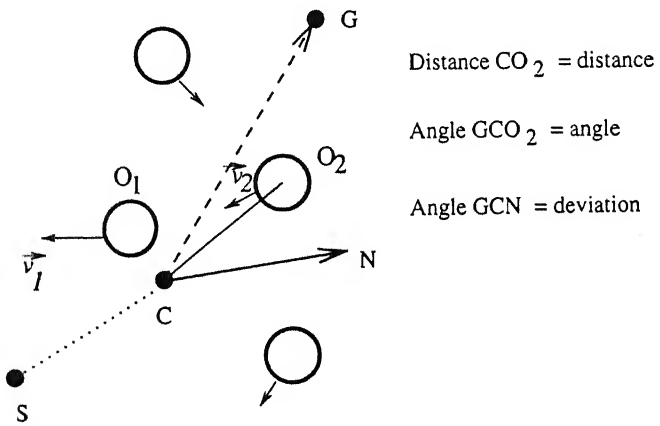


Figure 4.2: A schematic of condition (distance and angle) and action (deviation) variables.

angle, and relative velocity. The distance is the distance of the nearest obstacle forward

from the robot. Four fuzzy values of distance are chosen: very near (VN), near (NR), far (FR), and very far (VF). Each of them is assumed to take a triangular membership function as shown in Fig. 4.3. The angle is the relative angle between the path joining the robot and

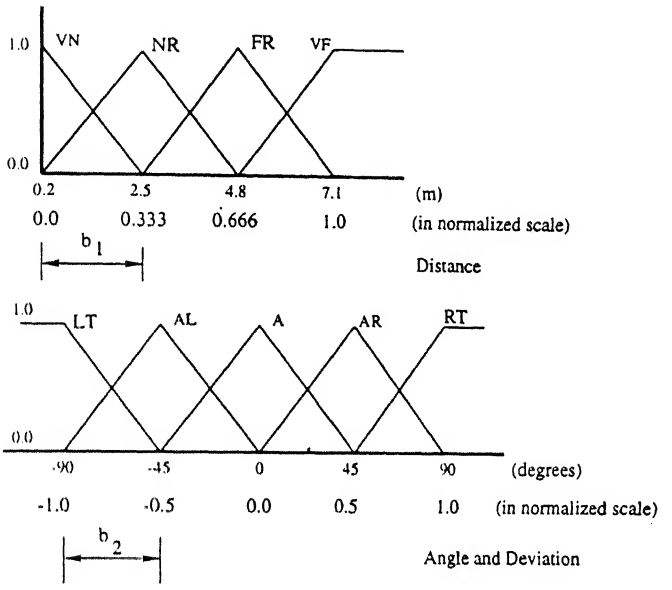


Figure 4.3: Author-defined membership functions for inputs and output of an FLC (navigation among moving obstacles).

the target point and the path to the nearest obstacle forward. The corresponding fuzzy values are left (LT), ahead left (AL), ahead (A), ahead right (AR), and right (RT). For simplicity, the shape of the membership function distributions is assumed to be triangular as shown in Fig. 4.3. The relative velocity is the relative velocity vector of the nearest obstacle forward with respect to the robot. Here, this information is not used explicitly as a fuzzy variable. Instead, a practical incremental procedure is adopted to eliminate its explicit consideration. In practice, the position and velocity of obstacles as a function of time may not be known a priori. However, the robot can find the position and velocity of each obstacle at a regular interval of time using sensors. Since at the end of each time step, the robot knows the position and relative velocity of each obstacle, the definition of the nearest obstacle forward can be modified by using the relative velocity information of obstacles. In such a case (Fig. 4.2), even if an obstacle O_1 is nearer compared to another obstacle O_2, and the relative velocity $\vec{v}_1$ of O_1 directs away from robot's path towards the target point G, whereas the relative velocity $\vec{v}_2$ of O_2 directs towards the robot (Position C), the obstacle O_2 is assumed to be the nearest obstacle forward. This

practical consideration helps to achieve two important things:

1. It eliminates the third condition variable (relative velocity) in the rule set. This reduces search space for finding optimal rule base considerably.

2. An incremental approach can be used, where the robot locates all obstacles at the end of a small time step. This makes the approach practical to be used in a real scenario.

The action variable is deviation of the robot from its path towards the target (Fig. 4.2). This variable is considered to have five fuzzy values: LT, AL, A, AR, and RT. The same triangular membership functions as those used for angle are used here. A typical rule will, thus, look like the following:

If distance is VN and angle is A, then deviation is AL.

With four choices for distance and five choices for angle, there could be a total of 4×5 or 20 combinations of two different conditions possible. For each of these 20 combinations, there could be one value of the action variable. Thus, there are a total of 20×5 or 100 different rules possible, but an arbitrary set from these 100 rules cannot be used to constitute a valid rule base. This is because for two rules having identical combination of condition variables, there should be a unique value of the action variable. Thus, the maximum number of rules that may be present in a rule base is 20, each having a unique combination of condition variables. All 20 rules which are used in this study are shown in Table 4.1. It is interesting to note that a particular value of the action variable for

Table 4.1: Author-defined rule base of an FLC (navigation among moving obstacles)

		angle				
		LT	AL	A	AR	RT
distance	VN	A	AR	AL	AL	A
	NR	A	A	AL	A	A
	FR	A	A	AR	A	A
	VF	A	A	A	A	A

each combination of condition variables has been assigned based on intuition. When an

obstacle is very near and straight ahead, the robot deviates towards ahead-left. However, when the obstacle is very near but on the left of the robot, the robot goes ahead. As the critical obstacle is away from the robot, it has a tendency to move ahead. This set of rule base is pretty good and it will be shown later that an FLC with this rule base can navigate well in certain scenarios.

It is important to note that not all 20 rules are necessary for the robot to use during an obstacle avoidance. One of the tasks in this study is to find which (and how many) rules should be there in the rule base for the robot to find the quickest path between two points. The presence and absence of a rule are represented by 1 and 0, respectively. Thus, a complete solution will have a 20-bit length string of 1 and 0. The value of the i-th position along the string marks the presence or absence of the i-th rule in the rule base. Thus, the following 20-bit string represents eight rules, as depicted in Table 4.2.

10011 01010 00010 01010

Thus, by using a 20-bit string, any combination of rules in the rule set can be represented.

Table 4.2: Eight rules represented by the above string

distance \ angle	LT	AL	A	AR	RT
VN	A			AL	A
NR		A		A	
FR				A	
VF		A		A	

4.2.2 Evaluating a Solution

A solution represented by a 20-bit string defines a set of rules which the robot uses to navigate through moving obstacles and attempts to reach point G (destination) from the point S (starting point). It is clear that some solutions will enable the robot to achieve the task quickly, whereas some will require longer time. In fact, not all solutions will necessarily take the robot to its destination within a fixed time (T_{max}). Thus, a solution is evaluated by calculating the total time (T) the robot takes to reach the destination.

If $T < T_{max}$, the solution is assigned a value of $f = T$. If, however, the time of travel exceeds T_{max} and the robot does not reach its destination, it is halted at its current position and the Eucledian distance d_{rem} between the halted position and the destination point is calculated. A value of $f = T_{max} + d_{rem}/v$ (where v is the maximum velocity of the robot) is assigned to this solution, approximately.

It is important to note that the above optimization process will largely depend on the particular scenario (the placement and motion of obstacles) used for the motion planning problem. Thus, in order to find a genetic-fuzzy solution which is fairly generic to a wide variety of scenarios, a solution is evaluated in H different scenarios and the average travel time (f above) of all H scenarios is used as the actual objective function value, which is minimized during the optimization process.

Now, the method adopted for calculation of the actual travel time T will be discussed in detail. As mentioned earlier, the robot's total path is a collection of a number of small straight line paths traveled for a constant time ΔT in each step. To make the matter as practical as possible, it is assumed that the robot starts from zero velocity and accelerates during the first quarter of the time ΔT and then maintains a constant velocity for the next one-half of ΔT and decelerates to zero velocity during the remaining quarter of the total time ΔT (refer to Fig. 3.2). The magnitudes of acceleration and deceleration are assumed to be equal. If this magnitude is a, then the total distance covered during the small time step ΔT is $3a\Delta T^2/16$. At the end of the constant velocity travel during each time step (ΔT), the robot senses the position and velocity of each obstacle and decides whether to continue moving in the same direction or to deviate from its path. This is achieved by first determining the predicted position of each obstacle, as follows:

$$P_{\text{predicted}} = P_{\text{present}} + (P_{\text{present}} - P_{\text{previous}}). \tag{4.1}$$

The predicted position is the linearly extrapolated position of an obstacle from its current position P_{present} along the path formed by joining the previous P_{previous} and present position. Thereafter, the nearest obstacle forward is determined based on $P_{\text{predicted}}$ values of all obstacles and fuzzy logic technique is applied to find the obstacle-free direction using the rule base dictated by the corresponding 20-bit string. The working principle of a fuzzy logic controller has been explained in Section 2.2.2. If the robot has to change its path, its velocity is reduced to zero at the end of the time step; otherwise the robot does not decelerate and continues in the same direction with the same velocity $a\Delta T/4$. It is interesting to note that when the latter case happens (the robot does not change its

course) in two consecutive time steps, there is a saving of $\Delta T/4$ sec in travel time per such occasion. Continuing in this fashion, when the robot comes closer to the destination and there is no critical obstacle in its way, the robot reaches its destination by starting its deceleration from a distance of $a\Delta T^2/32$. Overall time of travel (T) is then calculated by summing all intermediate time steps needed for the robot to reach its destination. This approach of robot navigation can be easily incorporated in a real-world scenario.

4.2.3 Optimizing for Minimum-Time Solution

The GA technique is used to find an optimal or a near-optimal obstacle-free path. Here, a binary-coded GA has been used. The GA operators and its working principle are described in Section 2.3.1. In short, the GA begins its search by randomly creating a number of solutions represented in binary-coded strings. Since solutions in this problem are represented in a 20-bit string, this problem is ideal to be solved using a GA. Each solution in the population is then evaluated to assign a *fitness* value. In this study, the fitness value is assigned as follows. Each solution (20-bit string) is evaluated to calculate a function value f_i for H different scenarios ($i = 1, 2 \ldots, H$) of moving obstacles. The fitness to the string is assigned as $FS = \left(\sum_{i=1}^{H} f_i\right)/H$. Since the objective is to minimize the overall travel time, a GA is used to find a string which corresponds to the minimum fitness value.

After each solution in the population is evaluated and fitness is assigned, the population is modified by using three operators—tournament selection, one-point crossover, and bit-wise mutation (discussed in Section 2.3.1). The tournament selection compares two solutions at a time from the population and chooses the solution having the smaller fitness value. Crossover operator exchanges bit information between two such strings obtained after tournament selection and creates two new strings (or solutions). The mutation operator compliments bit values at arbitrary places in a string to create a new string. After a new population of solutions are created, each of them is evaluated again to find a fitness value and all three operators are applied again. One iteration of these three operators followed by the evaluation procedure is called a *generation*. Generations proceed until a termination criterion is satisfied. In this study, a GA-run is taken until a pre-specified number of generations have elapsed.

4.3 Results

In this section, simulation results of the navigation problem of a mobile robot among moving obstacles are presented in a systematic manner (Pratihar et al. 1998b, 1999a, Deb et al. 1998). There are six different approaches studied here.

Approach 1: Author-defined fuzzy logic controller. There are three components of the FLC knowledge base, namely *scaling factors*, *membership functions* and *rule set*. It is important to note that both the scaling factors and membership functions taken together will represent the semantics of the symbols used by the FLC and the rule set represents the syntactic mapping among the symbols. In this approach, a fixed set of 20 rules (Table 4.1) and author-defined membership functions (Fig. 4.3) are used. Figure 4.3 shows that b_1 is set to 2.3 corresponding to a scaling factor for distance SF_d=6.9 and b_2 is set to 45 corresponding to a scaling factor for angle (deviation) SF_a=90.0. No optimization method is used to find optimal knowledge base of the FLC.

Approach 2: Tuning scaling factors of the state variables alone. A set of author-defined rule base is assumed (Table 4.1) and the tuning of scaling factors for condition and action variables is done keeping the relative spacing of membership distributions constant. The shape of the membership function is assumed to be triangular. The base coordinates of the membership functions are considered as variables. All 20 possible rules shown in Table 4.1 are used. Thus, the objective of this study is to tune the scaling factors for condition and action variables which, along with 20 rules presented in Table 4.1, will result in the obstacle-free path taking the smallest possible travel time between any two points. One parameter for each of the action variables is kept as the decision variable. The bases b_1 and b_2 (refer to Fig. 4.3) are coded in 10 bit substrings each, thereby making a GA string equal to 20 bits. The base b_1 is decoded in the range (1.0, 4.0) m and the base b_2 is decoded in the range (25.0, 60.0) degrees. Thus, the scaling factors are actually tuned, while leaving the term-set proportionally spaced. In all simulations here, the membership function distribution for deviation is kept the same as that in angle.

Approach 3: Tuning rule base alone. The membership functions are defined by the author, as shown in Fig. 4.3. The rule base is optimized in this study. The maximum

number of possible rules is 20. Here, the GA string is a 20-bit string (of 1 and 0 denoting presence or absence of rules, respectively) as illustrated earlier in Table 4.2. But the objective of this study is to search for those (and how many) rules from these 20 that will result in an obstacle-free path taking the smallest possible travel time between any two points.

Approach 4: Tuning scaling factors and rule base in stages. In this study, the optimized solutions obtained in Approaches 2 and 3 are combined together. Thus, the membership function used here is the same as that found in Approach 2 and the rule base is the same as that found in Approach 3.

Approach 5: Tuning scaling factors and rule base simultaneously. In this study, both optimization of finding optimized base width of triangular membership functions and finding an optimized rule base are achieved simultaneously. Here, a GA string is a 40-bit string with first 20 bits denoting the presence or absence of 20 possible rules, the next 10 bits are used to represent the base b_1 and the final 10 bits are used to represent the base b_2. Lower and upper bounds of these latter two variables are kept the same as those in Approach 2. The objective of this study is to find the size of triangular membership functions for condition and action variables and the rule base which will result in the obstacle-free path taking the smallest possible travel time between any two points. This is a more practical approach, since both optimizations are performed simultaneously.

Approach 6: Automatic two-stage design of fuzzy rules using GA. A set of good fuzzy rules are obtained, in this approach, through two stages. In the first stage, a GA-based learning technique is used to determine a set of fuzzy rules, whereas in the second stage, a GA-based tuning is adopted to further reduce the number of rules to be present in the rule base. There are two inputs (distance and angle) and one output (deviation) of the fuzzy logic controller. As four (VN, NR, FR, VF) and five (LT, AL, A, AR, RT) different values are assigned to distance and angle, respectively, a maximum of 20 rules are considered in the fuzzy rule base. The membership function distributions for the inputs and output are shown in Fig. 4.3 which are kept unaltered throughout the study. The purpose of this study is to find a set of good fuzzy rules using a GA. It is important to note that no time is spent on manual construction of fuzzy rule base, in this approach. The output - deviation is assumed to have five (LT, AL, A, AR, RT) different values similar to the input -

angle has. A binary-coded GA is used and 3 bits are assigned for determining the action variable for a particular set of condition variables. The first bit of every rule will indicate whether that particular rule is present or absent in the rule base (1 for presence and 0 for absence). Except the VF distance case, rest for all cases of distance, the output (deviation) will be different from the input (angle). Thus, for all rules (3×5 or 15) with distance less than or equal to FR, there are two more bits representing four different output (deviation) cases. Say, the input angle is A, then first bit represents presence of the rule and next two bits represent the action variable (100 for LT, 101 for AL, 110 for AR and 111 for RT). It is interesting to note that A is missing in the above list. This is because when the distance is less than or equal to FR, it does not make sense to move in the direction of the object. If the input angle is RT (for example), RT will be missing in the output deviation but other four, namely LT, AL, A and AR will be present and only two bits are needed to represent four options. In fact, other four cases (000, 001, 010 and 011) will indicate the absence of rule. For the VF case, just one bit is enough to indicate whether the rule is present or not (1 for presence and 0 for absence). Every time when the bit is 1, the output angle is A. Thus, we need $(15 \times 3 + 5 \times 1)$ or 50 bits to code the whole problem. Thus, a GA determines a set of good rules for an FLC through search. It is intuitive to say that the rule base obtained in the first stage will contain some redundant rules. This will happen due to the iterative nature of a GA. Thus, a second stage GA-based tuning of the rule base (obtained in the first stage) will further reduce the number of rules to be present in the rule base. This approach is called a *GA-based tuning of a GA-learned rule base*. The rule base obtained in this approach can be used for solving the unseen similar scenarios, on-line.

In order to investigate the efficacy of the proposed approaches, a scenario with only three moving obstacles is studied first. Thereafter, the results are presented for a more complicated scenario having eight obstacles. In all runs of the proposed approach, a binary tournament selection (with replacement), the single-point crossover operator with a probability p_c of 0.98 and the bit-wise mutation operator with a probability p_m of 0.02 are used. It is important to mention that the values of p_c and p_m are selected after a careful study with different sets of values of those parameters. The maximum number of generations is set to 40 in this study. In every case, a population size of 100 is used. In all the simulations here, time step ΔT and magnitude of acceleration (or deceleration) a are set to 4 sec and 1 m/sec^2, respectively. These values make the velocity of the robot

in the middle portion of each time step equal to 1 m/sec. In all approaches, $H = 10$ different author-defined scenarios (in which the size, initial position and velocity of the moving obstacles are kept different) are used to evaluate a solution. Fig. 4.4 shows the ten different training scenarios considered during the optimization phase for the three-obstacles case.

4.3.1 Three-Obstacles Problem

In this scenario, there are three obstacles moving independently in a grid of 16×12 m^2 in a 2-D space. The robot has to travel from point S to point G by avoiding all three obstacles. The results of all six proposed approaches are compared. The author-defined fuzzy logic controller (FLC) has all 20 rules (Table 4.1) and membership functions as shown in Fig. 4.3. The traveling distance and time are presented for all six approaches (Approaches 1 to 6) in Table 4.3. In this table, three (out of 10) scenarios used during the optimization process are shown in the first three rows. The subsequent three rows show three new (and different) scenarios which were not used during the optimization process. These three experiments show how the robot behaves in unknown scenarios. In

Table 4.3: Travel distance D (in m) and time T (in sec) obtained by six approaches for the three-obstacles problem

Sl.	Approach 1		Approach 2		Approach 3		Approach 4		Approach 5		Approach 6	
No.	D	T	D	T	D	T	D	T	D	T	D	T
1	16.18	17.57	15.43	17.32	14.61	16.48	14.61	16.48	14.61	16.48	14.17	15.89
2	15.57	17.51	14.89	16.85	14.32	16.09	14.32	16.09	14.32	16.09	14.25	15.99
3	16.98	19.64	16.50	20.00	14.80	17.74	14.80	17.74	15.46	18.35	14.58	17.44
4	18.92	23.92	18.16	22.91	14.67	16.57	14.67	16.57	14.67	16.57	14.67	16.57
5	15.80	18.79	15.43	18.31	14.36	14.58	14.36	14.58	14.36	14.58	16.36	19.82
6	17.09	20.79	16.25	19.67	14.59	16.46	14.59	16.46	14.59	16.46	14.28	16.04

most cases, Approach 2 is better than Approach 1 (with author-defined FLC). The fact that in some cases the performance in terms of time is almost the same reveals that the author-defined rule base contains good rules to find shorter paths in some scenarios. The table also shows that, in general, Approaches 3 - 6 have found better paths (in terms of travel time) than other two approaches. Importantly, the solution obtained in Approach 1

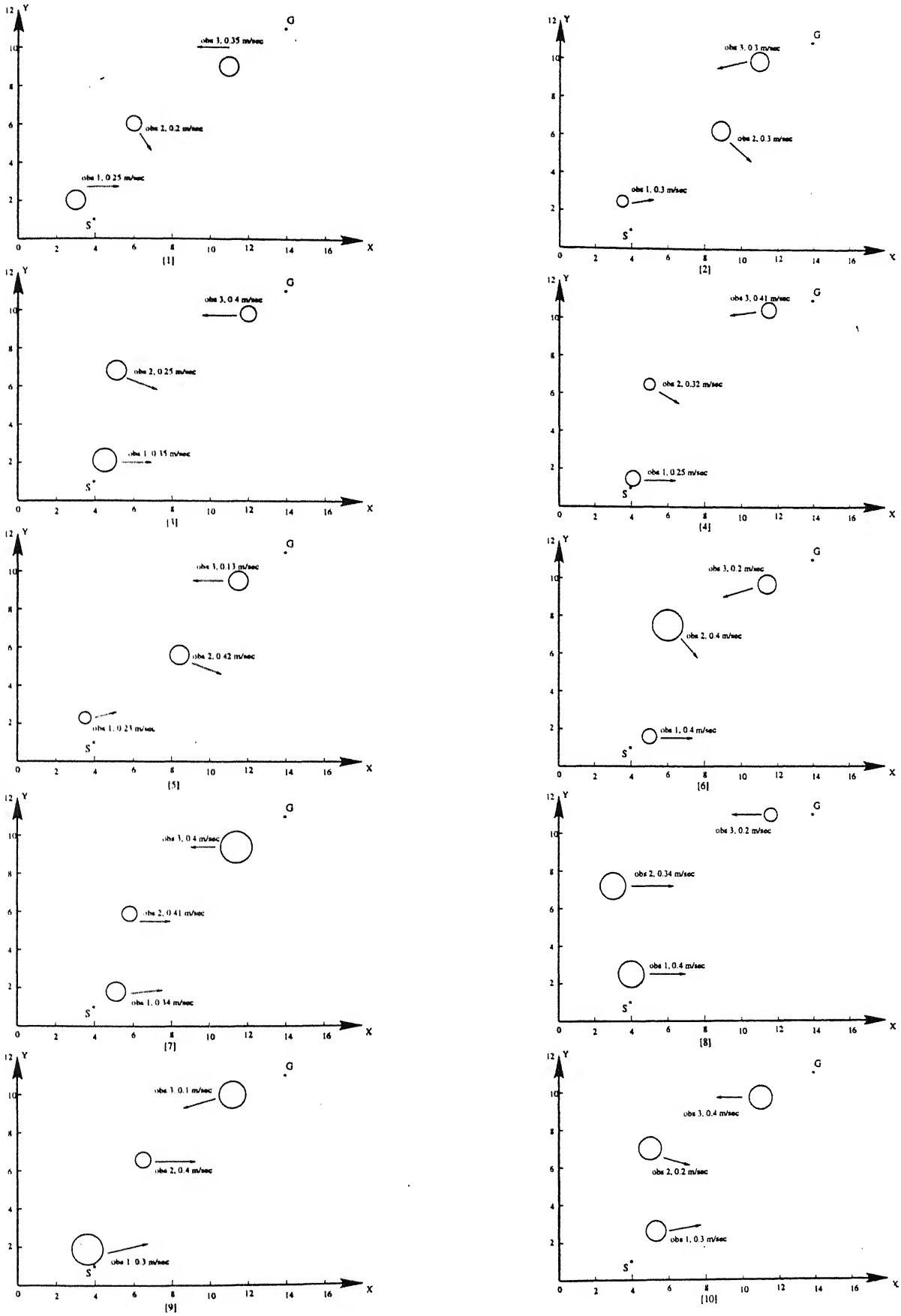


Figure 4.4: Training scenarios considered during the optimization phase - three-obstacles

(with no optimization) is not better than results obtained in Approaches 2 - 6 (except in scenario 5 in which Approach 1 is found to perform better than Approach 6). This happens because in scenario 5, the same obstacle has been treated as a critical obstacle more than once, in Approach 6.

The paths obtained in all six approaches for the test scenario 4 (presented in Table 4.3) are shown in Fig. 4.5. The corresponding travel distance and time are also shown in Table 4.3. It is clear that Approaches 3 - 6 have obtained a better path than the other two methods.

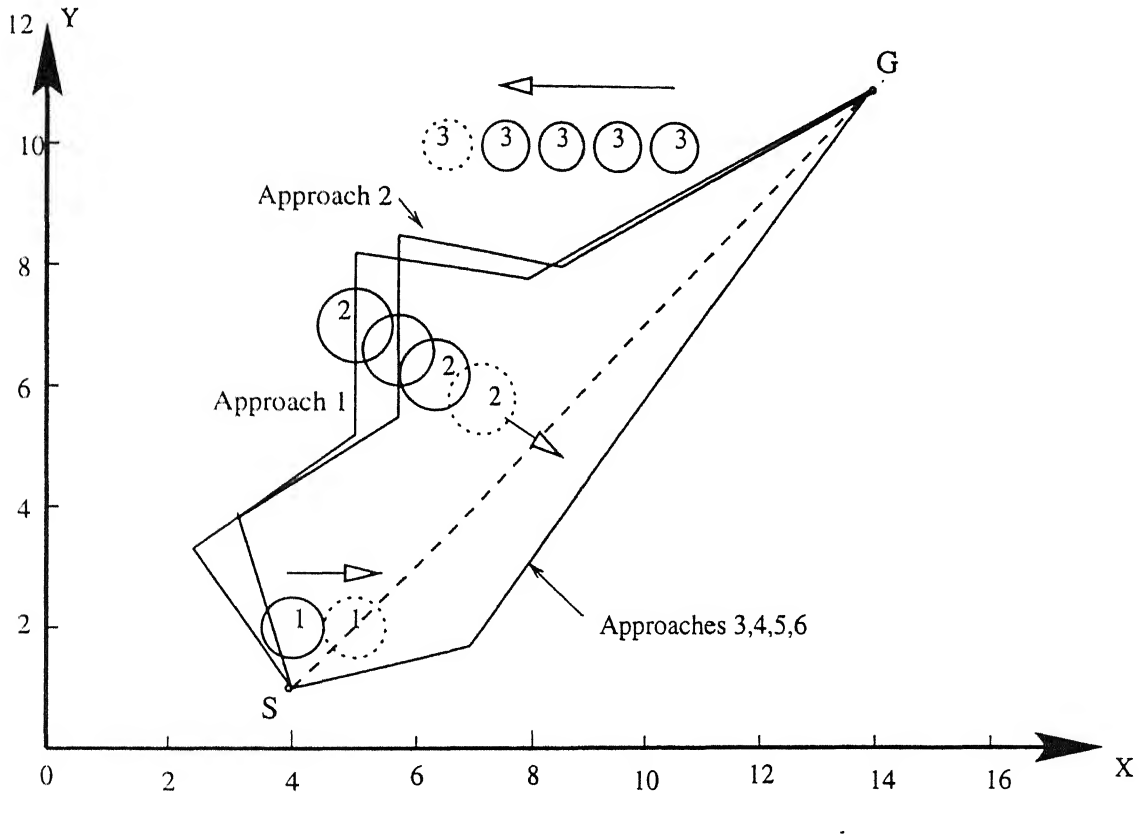


Figure 4.5: Optimized path found by all six approaches for the three-obstacles problem. The dashed circles mark the critical positions of obstacles.

The optimized rule base obtained using Approaches 3 to 5 are shown in Tables 4.4 and 4.5. These tables show that both rule bases are quite different from each other. There are only two rules those are common between the two rule bases. The optimized membership functions obtained using Approaches 2 and 5 are shown in Figs. 4.6 and 4.7, respectively. Fig. 4.6 shows that the scaling factors for distance (SF_d) and angle (SF_a)

Table 4.4: Optimized rule base (having six rules only) obtained using Approaches 3 and 4 for the three-obstacles problem

distance \ angle	LT	AL	A	AR	RT
VN					A
NR	A			A	
FR		A			
VF		A		A	

Table 4.5: Optimized rule base (having eight rules only) obtained using Approach 5 for the three-obstacles problem

distance \ angle	LT	AL	A	AR	RT
VN	A	AR			
NR	A				A
FR					
VF	A		A	A	A

are found to be 3.9 and 50.0, respectively (using Approach 2). Similarly, SF_d and SF_a are determined (using Approach 5) to be 1.9 and 50.8, respectively, as shown in Fig. 4.7. These figures show that although the optimized membership functions for angle and deviation are more or less similar, the optimized membership function for distance are very different in both cases. In Approach 5, the membership functions are squeezed so that in most situations, distances will appear as very far (VF). The corresponding rule base is adjusted so that there are more rules specifying rules related to very far distances (Table 4.5). It is interesting to note that although in some scenarios the same travel time is obtained, different approaches have used different combinations of rules and membership functions. Since the critical obstacle is always found away from the robot, most of the rules specifying an action for very far (VF) away obstacles are preferred. Moreover, Table 4.6 shows the optimized rule base obtained using Approach 6. It is important to note that the execution time of the proposed algorithm is negligible (less than 1.0 sec) in a HP 9000/K200 machine. Thus, it is suitable for on-line implementation.

Table 4.6: Optimized rule base (having six rules only) obtained using Approach 6 for the three-obstacles problem

distance \ angle	LT	AL	A	AR	RT
VN		LT	AR		
NR		A	AL		
FR			LT		
VF					A

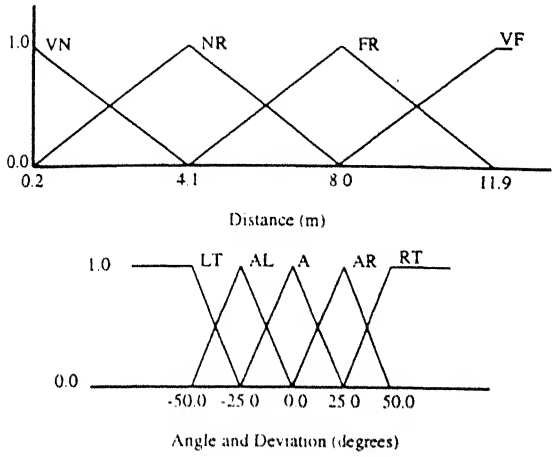


Figure 4.6: The optimized membership function obtained using Approach 2 for the three-obstacles problem.

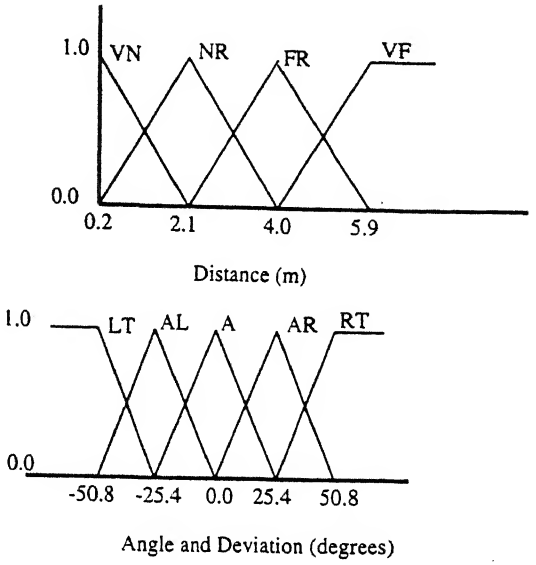


Figure 4.7: The optimized membership function obtained using Approach 5 for the three-obstacles problem.

4.3.2 Eight-Obstacles Problem

All six approaches have also been applied to eight-obstacles problems in a grid of 24×20 m^2. The optimized travel distance and time for Approaches 2 - 6 are presented in Table 4.7 along with those obtained for Approach 1 (once again, no optimization is used in this case, but the same 20 rules and membership function chosen in the three-obstacles problem are used). The first three rows in the table show the performance of all approaches on

Table 4.7: Travel distance D (in m) and time T (in sec) obtained by six approaches for the eight-obstacles problem

Sl.	Approach 1		Approach 2		Approach 3		Approach 4		Approach 5		Approach 6	
No.	D	T	D	T	D	T	D	T	D	T	D	T
1	27.20	28.90	26.08	27.77	26.15	27.87	26.15	27.87	26.15	27.87	26.00	27.66
2	26.96	28.94	25.97	27.62	26.03	26.55	26.03	26.55	26.03	26.55	25.95	26.76
3	29.85	36.80	28.62	35.16	26.66	34.55	26.66	34.55	27.14	35.00	27.63	31.59
4	33.46	43.36	26.40	27.91	26.24	27.51	26.24	27.51	26.24	27.51	26.24	27.51
5	32.84	41.78	27.13	33.00	26.54	32.39	26.54	32.39	27.04	33.00	26.53	32.37
6	33.46	43.36	28.00	31.33	27.16	31.00	27.16	31.00	27.16	31.00	27.16	31.00

scenarios that were used during the optimization process and the last three rows show their performance on new test (unseen) scenarios. The table shows that in all cases, Approaches 3 - 6 have performed better than the other two approaches. Once again, in most cases Approach 2 is better than Approach 1.

Paths obtained using all six approaches for scenario 4 (Table 4.7) are shown in Fig. 4.8. It is clear that the paths obtained by Approaches 3 - 6 are shorter and quicker than

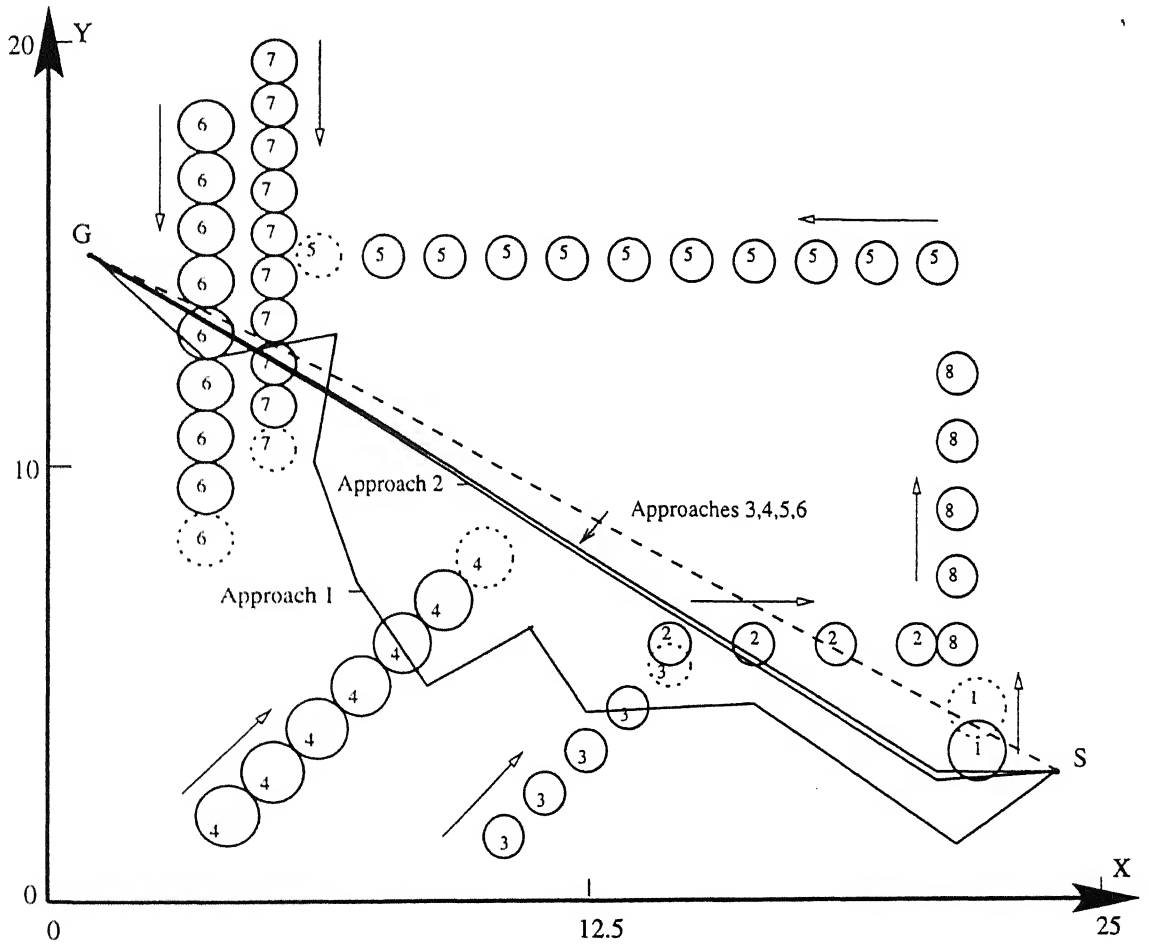


Figure 4.8: Optimized path found by all six approaches for the eight-obstacles problem. The dashed circles mark the critical positions of obstacles.

those obtained by Approaches 1 and 2.

The optimized rule bases obtained using Approaches 3 - 5 are shown in Tables 4.8 and 4.9. The optimized membership functions obtained using Approaches 2 and 5 are shown in Figs. 4.9 and 4.10, respectively. It is to be noted that a GA has found (using Approach

Table 4.8: Optimized rule base (having nine rules only) obtained using Approaches 3 and 4 for the eight-obstacles problem

distance \ angle	LT	AL	A	AR	RT
VN					
NR		A		A	
FR	A	A			
VF	A	A	A	A	A

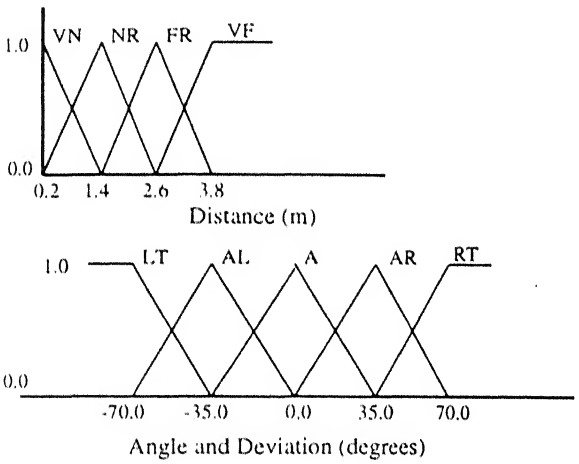


Figure 4.9: The optimized membership function obtained using Approach 2 for the eight-obstacles problem.

Table 4.9: Optimized rule base (having five rules only) obtained using Approach 5 for the eight-obstacles problem

distance \ angle	LT	AL	A	AR	RT
VN		AR			
NR	A	A			
FR				A	
VF				A	

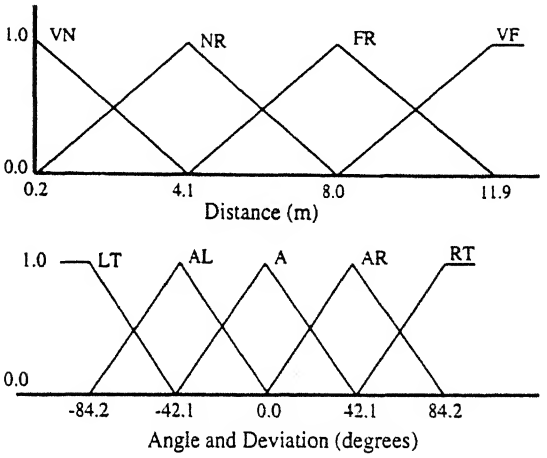


Figure 4.10: The optimized membership function obtained using Approach 5 for the eight-obstacles problem.

2) the values of scaling factors, namely SF_d and SF_a to be 1.2 and 70.0, respectively, as shown in Fig. 4.9. Similarly, Fig. 4.10 shows that the values of scaling factors, namely SF_d and SF_a are to be 3.9 and 84.2, respectively, as obtained by Approach 5. Here, Approach 5 (simultaneous tuning of rules and scaling factors) has elongated the base width of the triangle (representing membership function distribution) so that classification of relative angle is uniform in the range of $(-90, 90)$ degrees. Because only 10 scenarios are considered during the optimization process, it could have been that in most cases the critical obstacles come to the left of the robot, thereby causing more rules specifying LT or AL to appear in the optimized rule base. By considering more scenarios during the optimization process, such bias can be avoided and equal number of rules specifying left

and right considerations can be obtained.

In Tables 4.3 and 4.7, it can be observed that Approach 3 (tuning of rule base only) has resulted in a much quicker path than Approach 2 (tuning scaling factors of the state variables only). This is because finding a good set of rules is more important for the robot than finding a good set of membership functions. Thus, the optimization of rule base is a rough-tuning process and the tuning of base width of the triangle representing the membership function distribution is a fine-tuning process. In all cases, the optimized solutions are already obtained during the optimization of rule base only and tuning of scaling factors did not improve the solution any further.

Although the performance of Approaches 3 - 5 are more or less similar, Approach 5 is the most practical approach. In Approach 6, no time is spent on manual construction of a suitable rule base and the optimized rule base is obtained by a GA automatically. The optimized rule base obtained using Approach 6 is shown in Table 4.10. The results

Table 4.10: Optimized rule base (having six rules only) obtained using Approach 6 for the eight-obstacles problem

distance \ angle	LT	AL	A	AR	RT
VN	A	AR			
NR		A			
FR		AR			
VF		A			A

obtained by Approach 6 are comparable to those obtained by Approaches 3 - 5. Thus, Approach 6 is a more flexible approach for solving the navigation problems of a mobile robot among moving obstacles. Approach 4 is a two-stage optimization process, and hence require more computational effort. The similarity in the performances of Approaches 3 and 5 reveals that optimizing rule base has a significant effect and the tuning of scaling factors is only a secondary matter. For more number of obstacles and more complicated scenarios, a good guess of a rule base may not be possible. Thus, for more complicated problems, Approach 6 is recommended, since the optimized rule base of an FLC is obtained automatically using a GA. The execution time of the proposed algorithm is found to be negligible (less than 1.0 sec) in a HP 9000/K200 machine. Thus, it is suitable for on-line

implementation to solve the navigation problem of a mobile robot, in an optimal sense.

4.4 Summary

The working principle of a genetic-fuzzy approach (in which the performance of an FLC is improved by using a GA) has been explained in this chapter. The proposed genetic-fuzzy approach is used to solve the navigation problems of a mobile robot in the presence of moving obstacles. Here, a GA is used to find optimized FLC off-line and the obtained optimized FLC is used on-line, to solve the navigation problems of a mobile robot, in an optimal sense. It is observed that the performance of an FLC depends mainly on its rule base and optimizing membership function distribution is a fine tuning process. Thus, only two approaches, namely Approach 1 (Author-defined fuzzy logic controller) and Approach 3 (Tuning rule base alone) are considered in the subsequent studies (refer to Chapters 6-8). As some of the variables are discrete in nature, GA is an appropriate tool for solving this type of problems. Moreover, a method of automatic two-stage design of fuzzy rules has been developed by using a GA and its effectiveness has been tested for solving the navigation problems among moving obstacles. As no time is spent on manual construction of fuzzy rule base, this approach is recommended for solving a more complicated problem.

Chapter 5

PERIODIC GAIT GENERATION OF A LEGGED ROBOT

This chapter explains the gait generation algorithm, in detail and its effectiveness is tested through simulations for generating periodic gait of a six-legged robot.

5.1 Description of the Problem

The locomotion of a legged robot is characterized by its *gait* (refer to Section 1.3.2). The problem can be stated as follows: A six-legged robot (as considered in this study) will have to move on a flat terrain along a particular direction (straight path). Fig. 5.1 shows the schematic diagram of a six-legged robot. The reachable area for each leg is shown by the dotted square. The leg number is also assigned as indicated in the figure. In the periodic gait generation mode, the lifting or placing of legs will be done in a particular sequence which is fixed. Thus, the number of ground-legs is fixed while moving through a particular distance and following the periodic gait pattern. No optimization is carried out to further reduce the number of ground-legs. In this gait generation mode, only translation of the vehicle is to be considered. It is important to note that the stability margin should always be positive to ensure static stability of the vehicle. A few more terms related to the legged robot locomotion have been defined in the next sub-section (Song and Waldron 1989).

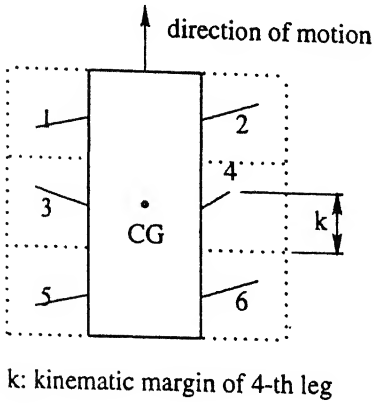


Figure 5.1: A schematic diagram of a six-legged robot.

5.1.1 A Few Definitions

1. *Transfer phase*: The transfer phase of a leg is the period during which the foot is in the air.

2. *Support phase*: The support phase of a leg is the period during which the foot is on the ground.

3. *Cycle time, t*: It is defined as the time for a complete cycle of leg locomotion of a periodic gait.

4. *Stride*: It is the distance traveled by the center of gravity of the vehicle during one complete locomotion cycle.

5. *Stroke*: It is defined as the distance through which the foot is translated relative to the body during the support phase.

6. *Pitch*: It is the distance between the centers of strokes of the adjacent legs on one side.

7. *Support pattern*: A support pattern is the convex hull of the vertical projections of the feet of all supporting legs of a multi-legged robot on a horizontal plane.

8. *Stability margin, s*: It is the shortest of the distances from the vertical projection of the center of gravity (CG) to the boundaries of the support pattern in the horizontal plane (refer to Fig. 5.3).

9. *Kinematic margin, k*: It is defined as the distance from the current foothold of leg i to the boundary of the reachable area of the leg i measured in the opposite direction of body motion (refer to Fig. 5.1).

10. *Duty factor, β*: It is defined as the time fraction of a cycle time in which leg i is in the support phase.

11. *Regular gait*: It is a gait with the same duty factor for all legs.

12. *Symmetric gait*: A gait is symmetric if the motion of the legs of any right-left pair is exactly half a cycle out of phase.

13. *Crab axis*: It is the axis which goes through the body center and is aligned with the direction of body motion.

14. *Crab angle*: It is defined as the angle from the longitudinal axis to the direction of motion.

15. *Deadlock situation*: It refers to a situation of a legged vehicle in which it is unable to maintain its static stability and there is no more leg in the air that can be placed on the ground to regain its stability. It is important to note that as the number of ground-legs increases, the chance of a deadlock situation occurring increases.

5.2 Mathematical Formulation of the Problem

The following assumptions are made to simplify the problem:

1. A terrain map can be built based on the available sensory information which is then discretized into a number of terrain cells. The center of each cell is considered as a candidate foothold.

2. The contact of the feet with the terrain (flat) can be modeled as point contacts. There is no slipping between the foot and the ground.

3. The mass of all the legs is lumped into the body and the center of gravity is assumed to be at the centroid of the body.

4. The body is held at a constant height and parallel to the ground plane during walking.

Two reference frames, namely world coordinate frame $\{W\}$ and body coordinate frame $\{B\}$, as shown in Fig. 5.2, have been considered for the purpose of analysis. The origin of the body coordinate frame is fixed at the center of gravity of the hexapod. Here,

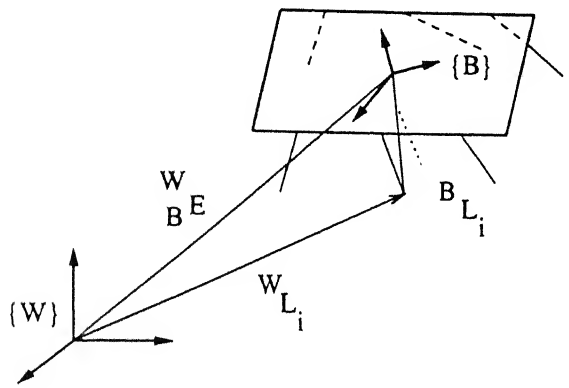


Figure 5.2: A schematic showing world frame and body frame.

${}^{W}_{B}E$ represents the transformation vector from $\{W\}$ to $\{B\}$ and ${}^{B}L_i$ indicates the position of i-th leg with respect to the body coordinate frame $\{B\}$. The position of i-th leg with respect to the world coordinate frame $\{W\}$ is represented by ${}^{W}L_i$ and ${}^{W}_{B}R$ indicates the rotation matrix. The position of a foot in the body coordinate frame is related to the position in the world coordinate frame as given by the expression:

$$ {}^{W}_{B}R\,{}^{B}L_i = {}^{W}L_i - {}^{W}_{B}E \tag{5.1} $$

It is important to mention that the rotation matrix, ${}^{W}_{B}R$ becomes a *unit matrix* when the hexapod moves along a straight path (only translation but no rotation), as considered here. The term ${}^{W}_{B}E$ can also be expressed as ${}^{W}G$, where ${}^{B}L_i$, ${}^{W}L_i$ and ${}^{W}G$ are expressed as follows: ${}^{B}L_i = \begin{pmatrix} {}^{B}X_i \\ {}^{B}Y_i \end{pmatrix}$, ${}^{W}L_i = \begin{pmatrix} {}^{W}X_i \\ {}^{W}Y_i \end{pmatrix}$, ${}^{W}G = \begin{pmatrix} {}^{W}G_X \\ {}^{W}G_Y \end{pmatrix}$.

It is assumed that the position of leg-i and the position of leg-j are different. Then, the margin s_{ij} can be represented in the body coordinate frame as follows:

$$ s_{ij} = \frac{{}^{B}X_i \times {}^{B}Y_j - {}^{B}Y_i \times {}^{B}X_j}{\sqrt{({}^{B}Y_i - {}^{B}Y_j) \times ({}^{B}Y_i - {}^{B}Y_j) + ({}^{B}X_i - {}^{B}X_j) \times ({}^{B}X_i - {}^{B}X_j)}} \tag{5.2} $$

Here, the ground leg-j follows another ground leg-i around the CG in the anti-clockwise direction so that the margin s_{ij} has a positive value only for the case of CG locating inside of the support polygon. Combining the equations (5.1) and (5.2), the margin s_{ij} can be expressed in the world coordinate frame as follows:

$$s_{ij} = \frac{U}{V}, \tag{5.3}$$

where

$$U = {}^{W}X_i \times {}^{W}Y_j - {}^{W}Y_i \times {}^{W}X_j + {}^{W}G_X \times ({}^{W}Y_i - {}^{W}Y_j) + {}^{W}G_Y \times ({}^{W}X_j - {}^{W}X_i),$$

and

$$V = \sqrt{({}^{W}Y_i - {}^{W}Y_j) \times ({}^{W}Y_i - {}^{W}Y_j) + ({}^{W}X_i - {}^{W}X_j) \times ({}^{W}X_i - {}^{W}X_j)}.$$

Thus, s_{ij} will have a positive value only when the CG of the body lies inside the support polygon which is a necessity for maintaining static stability of the vehicle. Fig. 5.3 shows the support pattern of a six-legged robot. The perpendicular distances from the projection of CG to all sides of the support pattern are determined and the minimum of those distances is considered as the stability margin, s. For a periodic gait (wave gait)

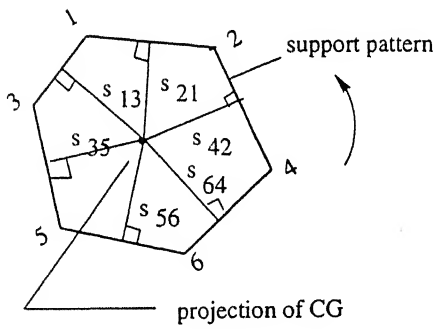


Figure 5.3: Stability margin of a support pattern.

with a duty factor β, the phases of legs 1,3,5,2,4,6 with respect to leg 1 are 0, β, $2\beta - 1$, $1/2$, $\beta - 1/2$, $2\beta - 1/2$, respectively (Pal et al. 1994). In the wave gait, the sequence of leg transfer is fixed for a particular value of duty factor, β. For fixed values of duty factor, $\beta(= 2/3)$ and phase difference, $\gamma(= 1/2)$, the sequence of leg transfer has been determined as shown in Fig. 5.4, which is to be maintained by the hexapod while generating its gait.

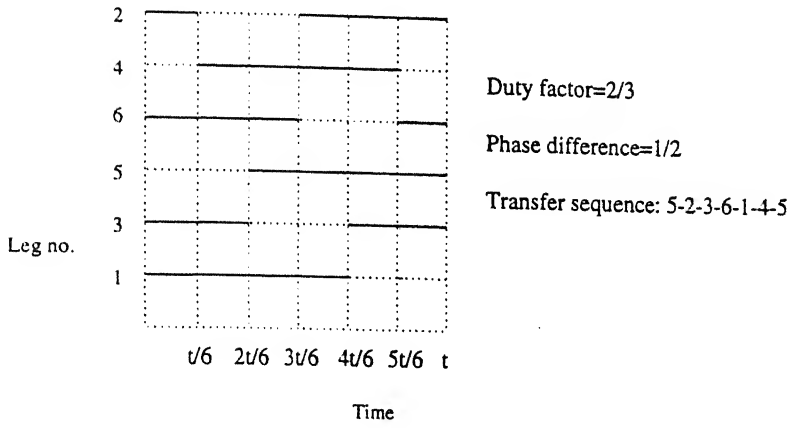


Figure 5.4: Gait-diagram (wave gait).

5.3 Methodology

A six-legged robot has to plan its gait (periodic) while moving on a flat terrain along a straight path. The total distance to be traveled is divided into Q equal parts usually known as *motion segments*. It is important to note that the decisions regarding lifting and placing of legs are taken at the end of each motion segment. The following steps are considered for designing the periodic gait of a hexapod:

1. The position of CG of the body is determined at each motion segment. In periodic gait generation, only translation of the vehicle is to be considered.

2. Each leg of the vehicle is controlled by a separate fuzzy logic controller (FLC). Thus, there are six FLCs to control the hexapod. As there is no scope to reduce the number of ground-legs further, particularly in this mode of gait generation, the output of each controller is assumed to be fixed and it is the stroke. In this study, the fixed leg-stroke is assumed to be equal to the distance traveled by the vehicle (CG of the body) in one motion segment.

3. Decision regarding the leg to be lifted to air, is taken based on the kinematic margin calculation at the immediately next and predicted support pattern (the support pattern just after transfer phase). If the kinematic margin of a particular leg is either zero or negative in the next support pattern, the leg is lifted to air.

4. The leg to be in transfer phase in a particular motion segment is selected based on

the gait diagram as shown in Fig. 5.4.

5. The vehicle is checked for its static stability. For static stability of the body, the CG should always lie inside the support polygon. If the stability is not maintained, decision regarding foot placement is taken. The leg on the air and with the highest positive kinematic margin is selected for placement on the ground, if required at all. There are some predefined candidate footholds for the placement of a leg on the ground.

6. If the stability (static) is maintained, the vehicle is allowed to move.

It is important to note that the same support pattern may continue only if the vehicle continues to remain stable and the supporting legs remain within their reachable area. Fig. 5.5 shows the flowchart of the gait generation algorithm.

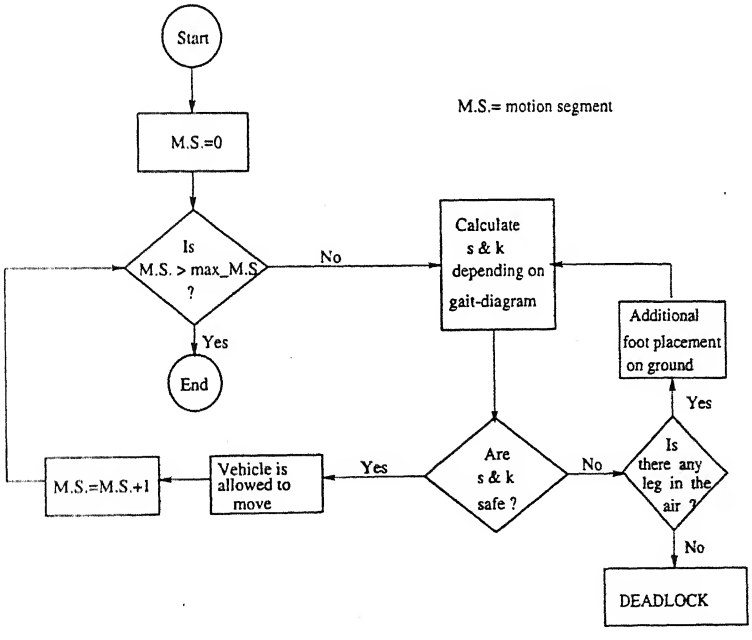


Figure 5.5: Flowchart of the gait generation algorithm.

5.4 Results

A six-legged robot (rectangular body plate with sides 0.9×0.4 m^2) will move along a straight path (only translation) on a flat terrain. Here, the total distance to be traveled by the hexapod is divided into $Q = 11$ (taken at random) motion segments. The path generated by the CG of the body is shown in Fig. 5.6. Fig. 5.7 shows the periodic gaits

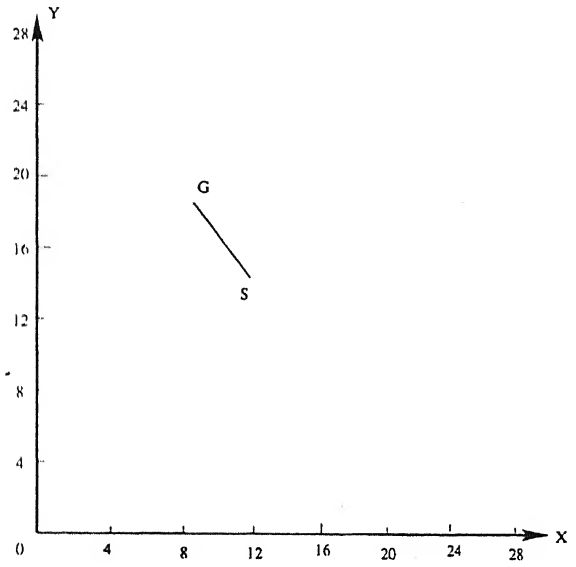
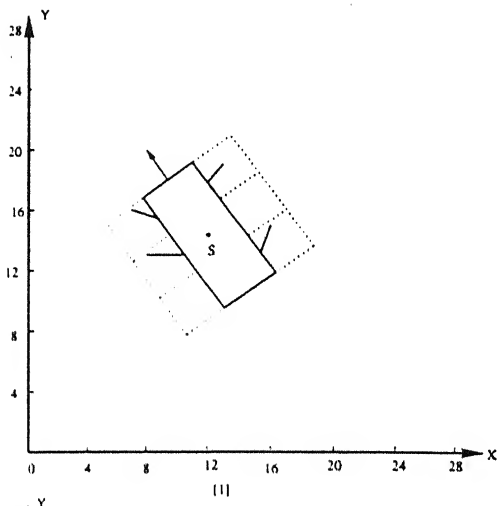
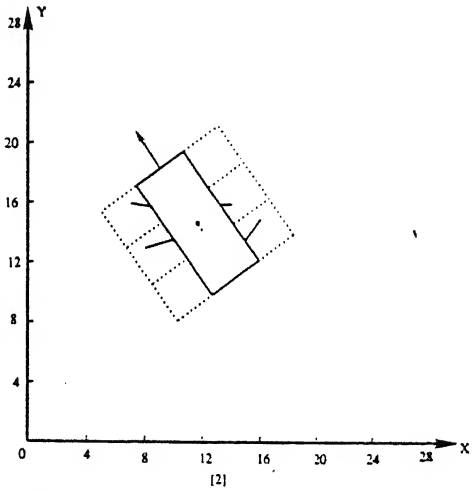
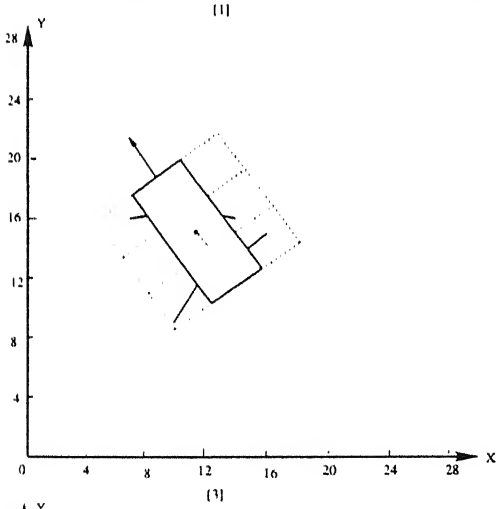


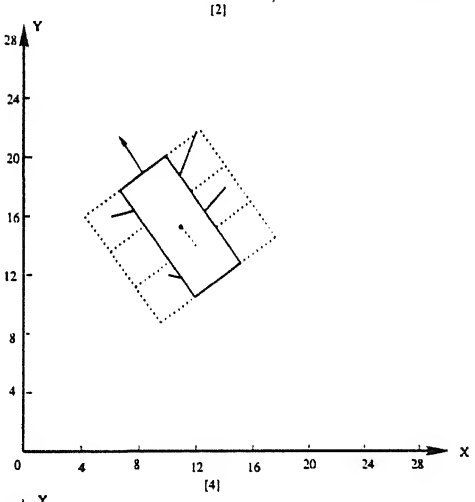
Figure 5.6: Movement of the CG of the vehicle while generating periodic gait.

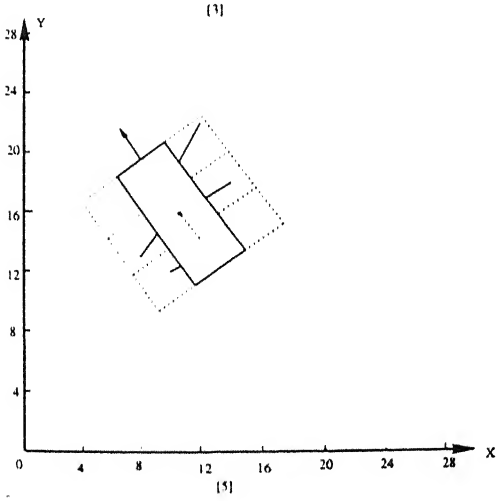
generated by the legged robot while moving from the initial point S to the final point G. It is important to note that in periodic gait generation, the sequence of leg transfer as well as the total number of ground-legs are fixed. Thus, no optimization is carried out here to further reduce the number of ground-legs.

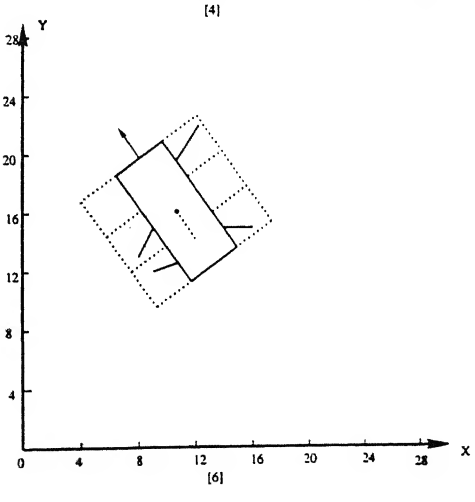

Y
X
S
0 4 8 12 16 20 24 28
4 8 12 16 20 24 28
[1]


Y
X
0 4 8 12 16 20 24 28
4 8 12 16 20 24 28
[2]


Y
X
0 4 8 12 16 20 24 28
4 8 12 16 20 24 28
[3]


Y
X
0 4 8 12 16 20 24 28
4 8 12 16 20 24 28
[4]


Y
X
0 4 8 12 16 20 24 28
4 8 12 16 20 24 28
[5]


Y
X
0 4 8 12 16 20 24 28
4 8 12 16 20 24 28
[6]

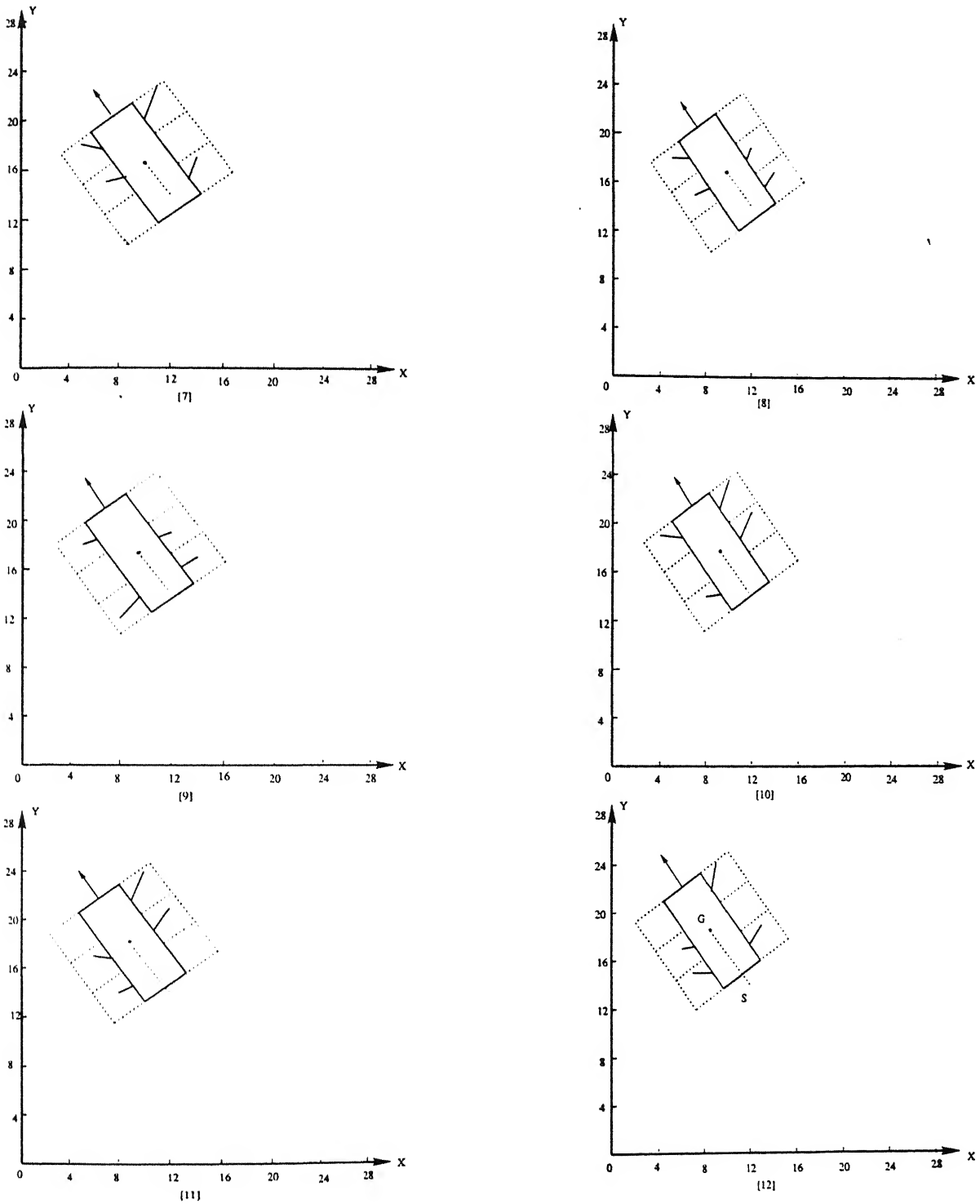


Figure 5.7: Periodic gait generation.

5.5 Summary

The gait generation algorithm has been discussed in this chapter. The effectiveness of the proposed algorithm is proved here for generating periodic gait of a six-legged robot through simulations. As the periodic gaits are optimal from the point of view of static stability, no optimization is carried out here to further improve the performance of the generated gaits.

Chapter 6

CRAB GAIT GENERATION OF A LEGGED ROBOT WHILE CROSSING A DITCH

In this chapter, a genetic-fuzzy approach has been used to generate optimal/near-optimal crab gaits of a six-legged robot while crossing a ditch. The six-legged robot will have to cross the ditch with minimum number of legs on the ground having the maximum average kinematic margin. A GA is used to find optimized FLCs off-line and the six legs of the robot are controlled by six different FLCs. The FLCs are computationally less expensive and the proposed algorithm is suitable for on-line implementation.

6.1 Problem Formulation

A six-legged robot (refer to Fig. 5.1) will have to cross a ditch while moving on flat terrain along a straight path (only translation is considered). The hexapod will move along the crab axis (refer to Section 5.1.1) and the generated gait is known as *crab gait*. The six-legged robot will cross the ditch with minimum number of ground-legs having the maximum average kinematic margin (refer to Section 5.1.1). Moreover, its stability margin (refer to Section 5.1.1) should always be positive to ensure static stability. It is to be noted that as the number of ground-leg increases, the probability of occurring a deadlock situation (refer to Section 5.1.1) increases. It is also interesting to note that the average kinematic margin of the ground-legs will indicate the potential progress of the

vehicle towards the goal. Thus, it can be treated as a constrained optimization problem.

The following assumptions are made to simplify the problem:

1. The terrain is discretized into cells and the center of each cell is considered as a candidate foothold.
2. The contact of the feet with the terrain (flat) can be modeled as point contacts and there is no slipping between the foot and the ground.
3. The mass of all the legs is lumped into the body and the center of gravity is located at the centroid of the body.

The gait planning problem of a six-legged robot (which is moving on flat terrain having a ditch) is considered here. Fig. 5.2 shows the two reference frames, namely world coordinate frame $\{W\}$ and body coordinate frame $\{B\}$ considered for the purpose of analysis. The origin of the body coordinate frame is assumed to be fixed at the center of gravity of the hexapod. Here, the crab walking of a hexapod has been studied where the body does not rotate but only translate along a certain direction. Here, ${}^{W}_{B}E$ represents the transformation vector from $\{W\}$ to $\{B\}$ and ${}^{B}L_i$ indicates the position of i-th leg with respect to the body coordinate frame $\{B\}$. The position of i-th leg with respect to the world coordinate frame $\{W\}$ is represented by ${}^{W}L_i$. As only translation of the body is to be considered here, the position of a foot in the body coordinate frame is related to the position in the world coordinate frame as given by the expression:

$$ {}^{B}L_i = {}^{W}L_i - {}^{W}_{B}E \tag{6.1} $$

The term ${}^{W}_{B}E$ can also be represented by ${}^{W}G$, where ${}^{B}L_i$, ${}^{W}L_i$ and ${}^{W}G$ are expressed as follows: ${}^{B}L_i = \begin{pmatrix} {}^{B}X_i \\ {}^{B}Y_i \end{pmatrix}$, ${}^{W}L_i = \begin{pmatrix} {}^{W}X_i \\ {}^{W}Y_i \end{pmatrix}$, ${}^{W}G = \begin{pmatrix} {}^{W}G_X \\ {}^{W}G_Y \end{pmatrix}$.

Moreover, the (stability) margin, s_{ij} can be calculated in the world coordinate frame $\{W\}$ using the expression:

$$ s_{ij} = \frac{U}{V}, \tag{6.2} $$

where

$$ U = {}^{W}X_i \times {}^{W}Y_j - {}^{W}Y_i \times {}^{W}X_j + {}^{W}G_X \times ({}^{W}Y_i - {}^{W}Y_j) + {}^{W}G_Y \times ({}^{W}X_j - {}^{W}X_i), $$

and

$$V = \sqrt{({}^{W}Y_i - {}^{W}Y_j) \times ({}^{W}Y_i - {}^{W}Y_j) + ({}^{W}X_i - {}^{W}X_j) \times ({}^{W}X_i - {}^{W}X_j)}.$$

Here, i and j are the positions of the feet of two supporting legs of a hexapod calculated in the anti-clockwise direction. Thus, s_{ij} will have a positive value only when the CG of the body lies inside the support polygon which is a necessary condition for maintaining static stability of the vehicle.

Here, the total distance to be traveled by the hexapod is divided into $Q = 9$ (taken at random) equal parts usually known as *motion segments*. Decisions regarding lifting and placing of legs are taken at the end of each motion segment.

The problem can be stated mathematically as follows:

$$\text{Maximize} \quad z = w_1 \times (6 \times Q - C) + w_2 \times K \tag{6.3}$$

subject to the conditions that the hexapod's leg does not fall on the forbidden zone and the stability margin, is restricted as

$$s > 0.$$

Q is the total number of motion segments, C is total number of ground-legs in Q motion segments, K indicates the average kinematic margin of the ground-legs, s (minimum of the s_{ij} values) indicates the stability margin of the vehicle, w_1 and w_2 are the weighting factors. It is important to mention that the constraint of stability margin is handled with the help of a penalty function approach.

6.2 Proposed Algorithm

In the proposed genetic-fuzzy system, the optimized FLCs are found by using a GA. The performance of an FLC depends on its rule base and membership function distributions. It is observed that optimizing rule base is a rough tuning process, whereas optimizing membership function distribution is a fine tuning process (Deb et al. 1998, Pratihar et al. 1998b, 1999a). Thus, only the rule base of an FLC has been optimized, in this study. Fig. 4.1 shows a genetic-fuzzy system. Here, the state of ditch is fed as input to the FLCs

to generate the optimal crab gait of a six-legged robot while crossing a ditch. Each leg of the hexapod is controlled by a separate fuzzy logic controller. Thus, there are six FLCs working in parallel. Fig. 6.1 shows the inputs of an FLC. A hypothetical intersection point, E (refer to Fig. 6.1) of the line joining the two extreme points of the ditch and the crab axis has been determined. There are two inputs (distance and relative angle) and one output (leg stroke) of the fuzzy logic controller. The distance is the distance of the

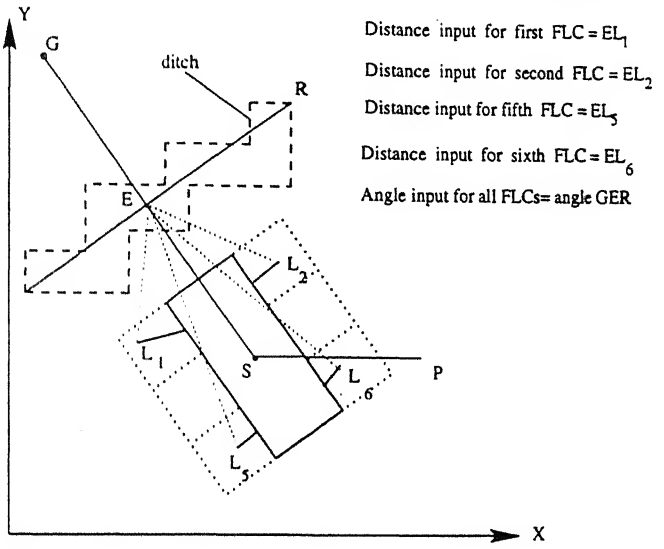


Figure 6.1: A schematic showing inputs of an FLC (ditch crossing module).

foot of a particular ground-leg from the intersection point E. The relative angle is the angle between the line joining the two extreme points of the ditch and the crab axis. The term leg stroke has been defined earlier (refer to Section 5.1.1). The proposed algorithm is based on the *stroke control strategy*. There are four (VN, NR, FR, VF) and five (NL, NM, Z, PM, PL) different values for distance and relative angle, respectively. Moreover, the output (stroke) has six (VS, SM, MM, SL, LR, VL) different values. The membership function distributions for input and output used in this study (author-defined) are shown in Fig. 6.2. For simplicity, the shape of the membership function distributions is assumed to be triangular. As there are four and five different values of distance and relative angle, respectively, 20 (4×5) rules are considered for each FLC. Thus, 120 rules are considered for all the six FLCs. The author-defined rule base for an FLC is shown in Table 6.1 and it is the same for all the six FLCs. Thus, a typical fuzzy rule will look as follows:

If distance is *VN* and relative angle is *NM*, then stroke is *SL*.

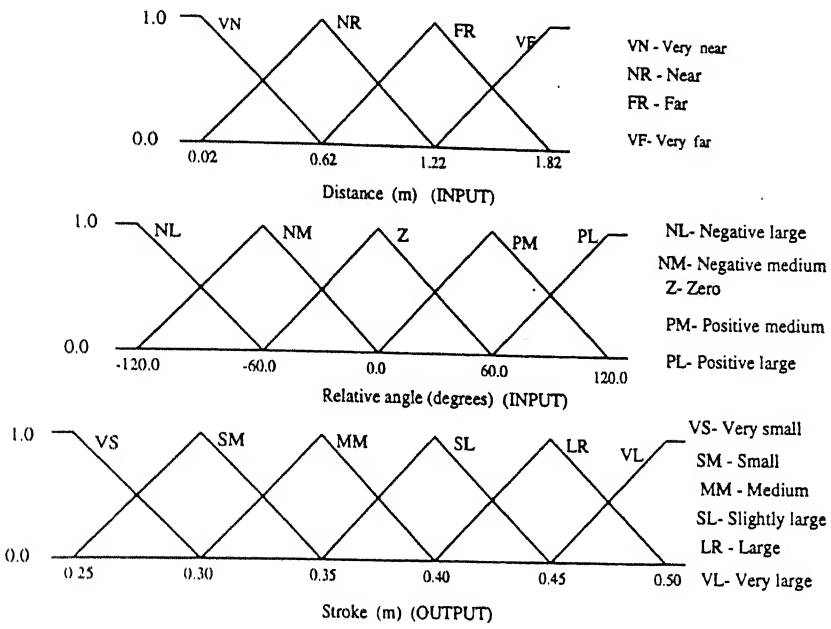


Figure 6.2: Author-defined membership function distributions for input and output of an FLC (crab gait).

The following steps are considered for designing the crab gait of a hexapod while crossing a ditch:

1. The position of CG of the body is determined at each motion segment. It is to be mentioned that the movement of the body is the same with that of the CG, for a straight-line motion.
2. The stroke for each ground-leg is determined using a fuzzy logic controller.
3. Decision regarding the leg to be lifted to air, is taken based on the kinematic margin

Table 6.1: Author-defined rule base for each FLC (crab gait)

distance \ relative angle	NL	NM	Z	PM	PL
VN	MM	SL	LR	SL	MM
NR	SM	SL	LR	SL	SM
FR	SM	MM	LR	MM	SM
VF	SM	SM	SL	SM	SM

calculation at the immediately next and predicted support pattern (the support pattern just after transfer phase). If the kinematic margin of a particular leg is either zero or negative in the next support pattern, the leg is lifted to the air.

4. At the beginning of a particular motion segment, if the kinematic margin of a leg is either zero or negative, it is selected as a transfer-leg.

5. The vehicle is checked for its static stability. The CG should always lie inside the support pattern to ensure the static stability of the body. If the stability is not maintained, decision regarding foot placement is taken. The leg on the air and with the highest positive kinematic margin is selected for placement on the ground, if required at all. There are some predefined candidate footholds for the placement of a leg on the ground.

6. The vehicle is allowed to move, only if its stability is maintained.

The same support pattern may also continue if the vehicle maintains its stability and the supporting legs remain within their reachable area. The step-worthiness of the terrain is also checked to ensure that the vehicle does not fall on the ditch. The maximum width of the ditch that can be crossed by the hexapod is determined from its leg-dimensions.

6.2.1 GA Representation of a Solution and Fitness Evaluation

Here, a binary-coded GA (refer to Section 2.3.1) with 120-bit string is used to represent a solution. The GA string will look as follows:

$$\underbrace{101\ldots1}_{\text{first FLC}}\ \underbrace{011\ldots0}_{\text{second FLC}}\ \underbrace{100\ldots1}_{\text{third FLC}}\ \underbrace{001\ldots0}_{\text{fourth FLC}}\ \underbrace{011\ldots1}_{\text{fifth FLC}}\ \underbrace{100\ldots0}_{\text{sixth FLC}}$$

Here, 1 and 0 represent the presence and absence of fuzzy rule, respectively. Each solution is then evaluated to calculate a function value z_i (refer to equation 6.3) for H different scenarios ($i = 1, 2 \ldots, H$). The fitness of the string is assigned as $FS = \left(\sum_{i=1}^{H} z_i\right)/H$. Here, a binary-coded GA is used to find a string which corresponds to the maximum fitness value. It is important to mention that the optimization is done off-line. Once the optimal rule base is obtained, the proposed algorithm is suitable for on-line implementation to determine the optimal gait.

6.3 Simulation Results and Discussion

The performance of a GA depends on the selection of its parameters. The best performance is found with the following GA parameters:

Population size = 100
Crossover probability = 0.9
Mutation probability = 0.01
Number of generations = 50

To show the effectiveness of the proposed algorithm, simulation results of the gait generation problem of a hexapod crossing a ditch are presented here (Pratihar et al. 1998a, Pratihar et al. 1999b). Two different approaches have been studied here.

Approach 1: Author-defined fuzzy logic controller. In this approach, a fixed set of 120 rules and author-defined membership functions (Fig. 6.2) are used. Table 6.1 shows a set of 20 author-defined rules used in one FLC and the same rule base is used in all the six FLCs. Thus, there is a fixed set of 120 rules. No optimization method is used to find the optimal rule base or to find the optimal membership function distributions.

Approach 2: Optimizing rule base alone. In this study, the rule base is optimized keeping the membership function distributions same as shown in Fig. 6.2. The maximum number of possible rules is 120. Here, a binary-coded GA with 120-bit string is used in which 1 and 0 indicate presence and absence of rules, respectively. Thus, a GA will find those (and how many) rules from the 120 rules that will result in a situation in which a hexapod will cross a ditch with minimum number of ground-legs and with maximum average kinematic margin after satisfying the constraint of stability margin. In this approach, H=10 different author-defined scenarios (in which the shape, size and orientation of the ditch are kept different) are used to evaluate a solution. Moreover, the weighting factors w_1 and w_2 are set to 1.0 and 5.0, respectively, after a careful study.

The gait generation problem is solved using both the approaches mentioned above.The minimum number of ground-legs and the maximum average kinematic margin are pre-

sented for the two approaches in Table 6.2. In this table, three scenarios (out of 10) used

Table 6.2: Number of ground-legs, C and average kinematic margin of ground-legs, K obtained by two approaches (crab gait)

Scenario	Approach 1		Approach 2	
	C	K	C	K
1	36	1.36	35	1.48
2	37	1.38	36	1.50
3	37	1.43	37	1.54
4	37	1.38	36	1.50
5	38	1.39	37	1.51
6	36	1.36	35	1.48
7	-	-	37	1.43

during the optimization process are shown in the first three rows. The subsequent four rows show four different and new scenarios which were not used during the optimization process. In scenario 7 (Table 6.2), it is seen that the author-defined FLCs have failed to generate stable gaits for the hexapod, whereas the GA-designed FLCs have successfully generated the stable gaits while crossing a ditch. In all cases, the GA-designed FLCs are found to perform better than the author-defined ordinary FLCs. The optimized rule base obtained through approach 2 for first, second, third, fourth, fifth and sixth FLCs are shown in Tables 6.3 through 6.8, respectively. It is to be mentioned that the presence of rules in the optimum rule base for each FLC depends on the number and nature of training cases considered. It is intuitive to say that if more number of training cases (symmetric in nature) are considered, the optimum rule base of first and second FLCs, third and fourth FLCs, and fifth and sixth FLCs will be symmetric in nature. It is also important to mention that due to the iterative nature of a GA, the obtained (GA-tuned) rule bases may contain some redundant rules and a second stage GA-based tuning will further reduce the number of rules to be present in the optimized rule bases. The generated gaits obtained using Approach 2 for the test scenarios 4 and 7 (Table 6.2) are shown in Figs. 6.3 and 6.4, respectively. It is important to mention that there must be close relationship between the length of the legs and the width of the ditch to be crossed. A six-legged robot having smaller legs cannot cross a very wide ditch. Thus, these parameters are to be selected very carefully in the simulations.

Table 6.3: Optimized rule base for first FLC (having nine rules only) obtained using Approach 2 (crab gait)

distance \ relative angle	NL	NM	Z	PM	PL
VN			LR		MM
NR	SM	SL	LR		
FR		MM			
VF	SM			SM	SM

Table 6.4: Optimized rule base for second FLC (having twelve rules only) obtained using Approach 2 (crab gait)

distance \ relative angle	NL	NM	Z	PM	PL
VN	MM			SL	MM
NR	SM	SL	LR	SL	SM
FR				MM	SM
VF	SM		SL		

Table 6.5: Optimized rule base for third FLC (having twelve rules only) obtained using Approach 2 (crab gait)

distance \ relative angle	NL	NM	Z	PM	PL
VN	MM		LR	SL	MM
NR				SL	SM
FR		MM	LR	MM	SM
VF		SM	SL		

Table 6.6: Optimized rule base for fourth FLC (having ten rules only) obtained using Approach 2 (crab gait)

distance \ relative angle	NL	NM	Z	PM	PL
VN		SL			MM
NR		SL			SM
FR	SM		LR	MM	
VF			SL	SM	SM

Table 6.7: Optimized rule base for fifth FLC (having twelve rules only) obtained using Approach 2 (crab gait)

distance \ relative angle	NL	NM	Z	PM	PL
VN		SL		SL	MM
NR	SM	SL	LR		
FR		MM	LR	MM	
VF		SM	SL	SM	

Table 6.8: Optimized rule base for sixth FLC (having eight rules only) obtained using Approach 2 (crab gait)

distance \ relative angle	NL	NM	Z	PM	PL
VN	MM			SL	MM
NR					
FR	SM	MM			SM
VF	SM		SL		

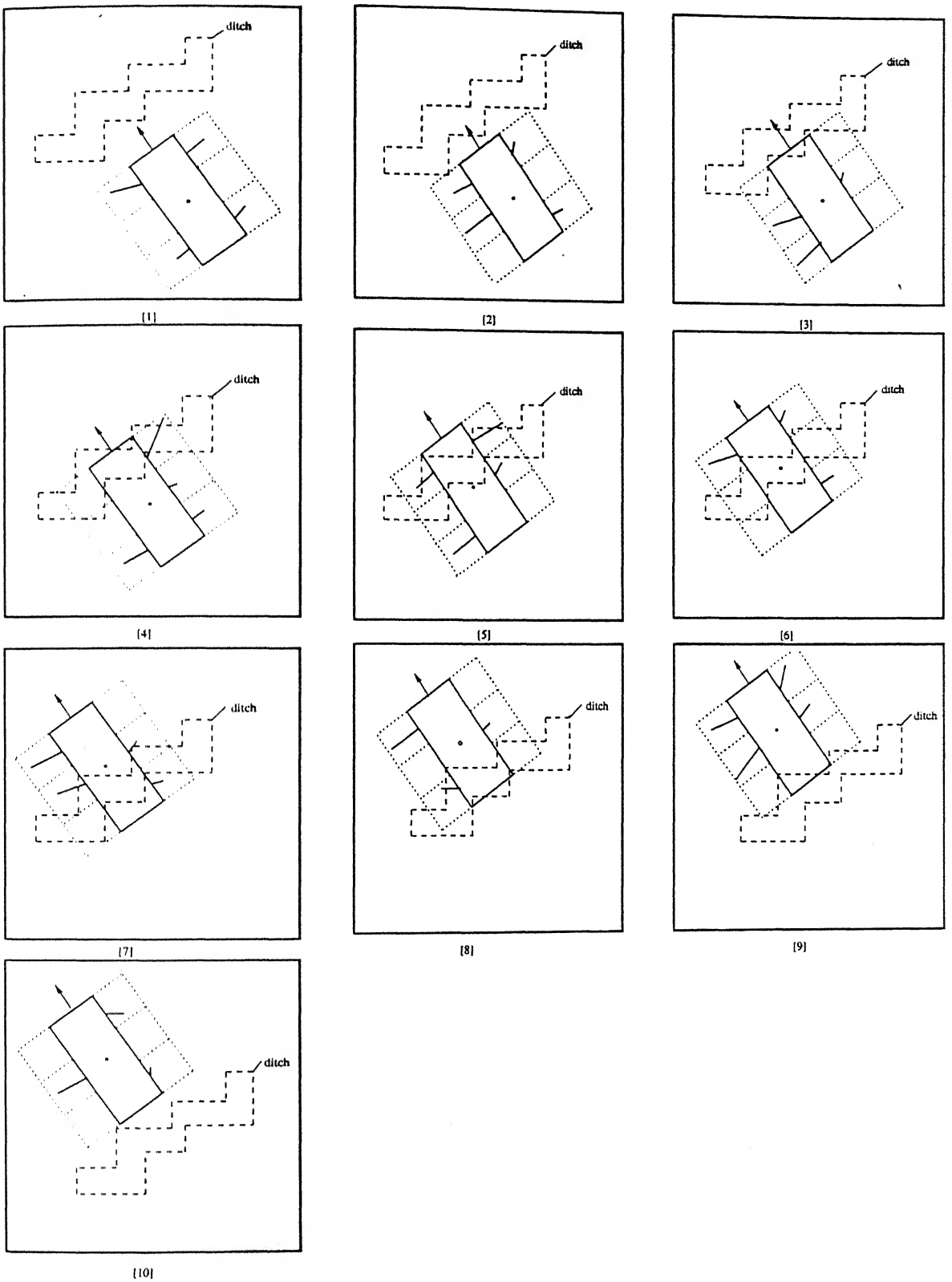


Figure 6.3: Generated gaits obtained using Approach 2 for test scenario 4 (Table 6.2).

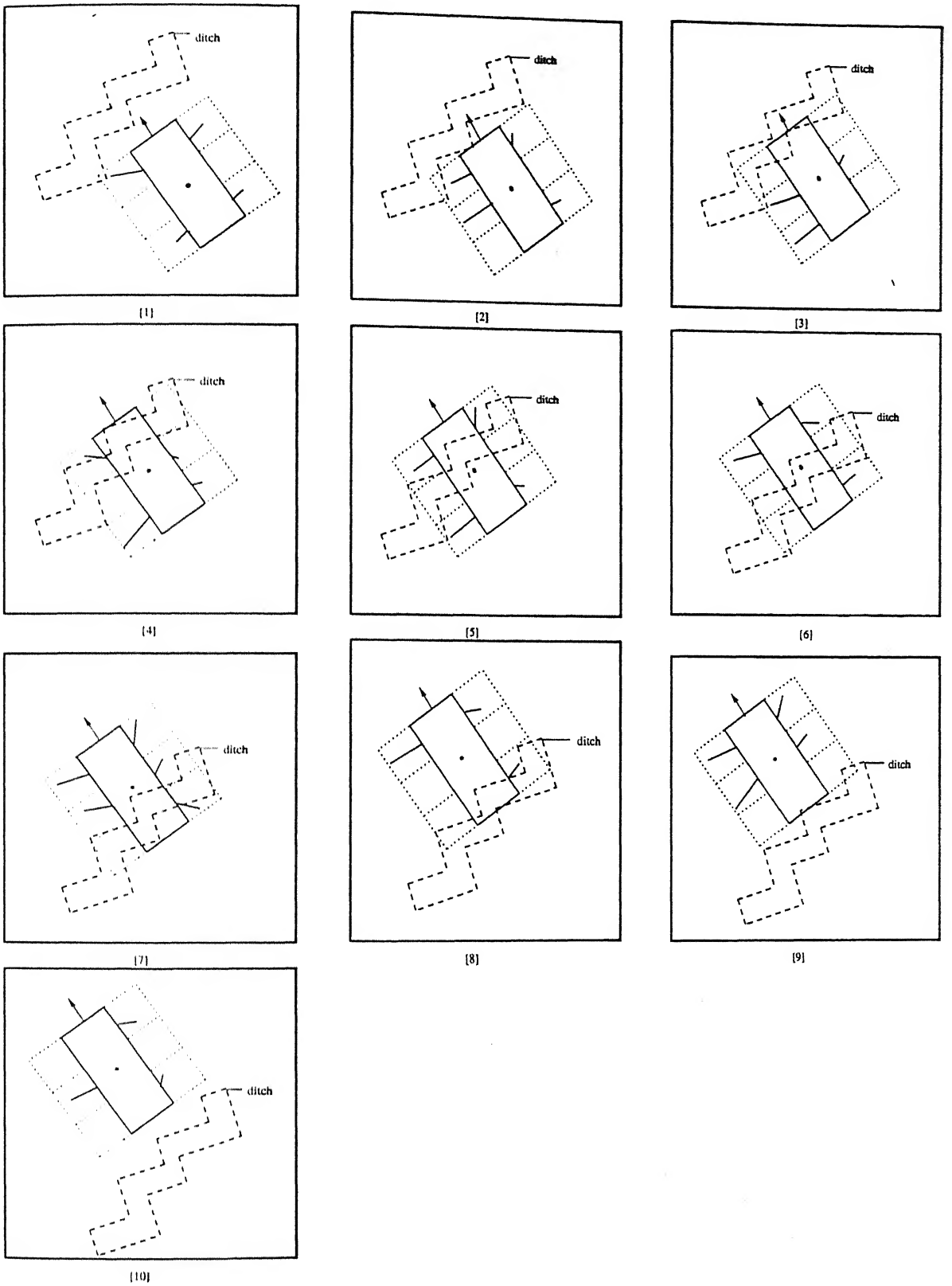


Figure 6.4: Generated gaits obtained using Approach 2 for test scenario 7 (Table 6.2).

6.3.1 Comparison of the Proposed Algorithm with the Graph Search Technique

Pal et al. (1994) used a graph search technique along with A* algorithm to solve the gait planning problem of a six-legged robot while crossing a ditch. The graph search technique has the following limitations:

- It is not suitable for on-line implementation due to its high computational load.
- The generated gait may be locally optimal.

In the proposed algorithm, GA-tuned FLCs are obtained off-line and then the optimized FLCs are used on-line, to solve the gait generation problem of a six-legged robot. The execution time of the proposed algorithm is found to be less than 1.0 sec in a HP 9000/K200 machine. Thus, it is suitable for on-line implementation to generate optimal/near-optimal gaits. As GA is a population-based search technique, there is a chance of obtaining a globally optimal solution. Thus, the proposed algorithm may overcome the limitations of the graph search technique.

6.4 Summary

In this chapter, a genetic-fuzzy system (in which a GA is used to improve the performance of the FLCs) is used to generate optimal/near-optimal crab gaits of a six-legged robot while crossing a ditch. There are six FLCs running in parallel, to control the six legs of the robot. The optimized FLCs are obtained off-line, by using a binary-coded GA. Once the optimized FLCs are available, the hexapod can use them on-line, to generate optimal/near-optimal gaits. Rule base (of an FLC) optimization involves the problem of dealing with discrete variables and GA is a potential tool for solving this type of problems. As GA is a population-based search and optimization technique, the chance of its solutions for getting trapped into local minima is less. Moreover, the proposed algorithm is found to be computationally tractable. Thus, it may overcome the drawbacks of the graph search technique (Pal et al. 1994) used for solving the gait planning problem of a legged robot.

Chapter 7

TURNING GAIT GENERATION OF A LEGGED ROBOT

A genetic-fuzzy approach is used here to generate optimal/near-optimal turning gait of a six-legged robot. The hexapod will have to take a circular turn through some angle in either clockwise or anti-clockwise direction in an optimal sense (with minimum number of ground-legs having the maximum average kinematic margin). A GA is used to find optimized FLCs off-line and these optimized FLCs (six different FLCs for the six legs of the robot) can be used on-line, to solve the problem of turning gait generation of a legged robot, in an optimal sense.

7.1 Statement of the Problem

A six-legged robot (shown in Fig. 5.1) will have to take a sharp circular turn in either clockwise or anti-clockwise direction with minimum number of legs on the ground and with maximum average kinematic margin (Section 5.1.1) of the ground-legs. Moreover, its stability margin (Section 5.1.1) should always be positive which is a necessary condition for maintaining static stability of the vehicle. In turning gait mode, both translation as well as rotation of the vehicle are to be considered. One of the difficulties usually faced by a multi-legged robot during its locomotion is the chance of occurring of a deadlock situation (Section 5.1.1). It is important to note that this chance will increase as the number of ground-leg increases. Moreover, the more the value of average kinematic margin of

the ground-legs, the more will be the progress of the vehicle towards the goal. Thus, the hexapod is required to take the circular turn with minimum number of ground-legs having the maximum average kinematic margin. It can be considered as a constrained optimization problem.

In practice, the vehicle will gather information of the surroundings by using different types of sensors. It is assumed that the terrain (flat) is discretized into a number of cells called *terrain cells* and the center of each cell is treated as a candidate foothold. Moreover, a point contact (with no slipping) is assumed to be present between the foot of the robot and the terrain. The mass of all the legs is lumped into the body and the center of gravity coincides with the centroid of the body.

Fig. 5.2 shows the two reference frames, namely world coordinate frame $\{W\}$ and body coordinate frame $\{B\}$, considered for the purpose of analysis. The origin of the body coordinate frame coincides with the center of gravity of the vehicle. As both translation as well as rotation of the vehicle are to be considered here, the position of a foot in the body coordinate frame is related to the position in the world coordinate frame as given by the expression:

$$ {}^{W}_{B}R\,{}^{B}L_i = {}^{W}L_i - {}^{W}_{B}E \tag{7.1} $$

The different terms have been explained in Section 5.2. The (stability) margin of the vehicle, s_{ij} can be expressed in the world coordinate frame as follows:

$$ s_{ij} = \frac{U}{V}, \tag{7.2} $$

where

$$ U = {}^{W}X_i \times {}^{W}Y_j - {}^{W}Y_i \times {}^{W}X_j + {}^{W}G_X \times ({}^{W}Y_i - {}^{W}Y_j) + {}^{W}G_Y \times ({}^{W}X_j - {}^{W}X_i), $$

and

$$ V = {}^{W}_{B}R\sqrt{({}^{W}Y_i - {}^{W}Y_j) \times ({}^{W}Y_i - {}^{W}Y_j) + ({}^{W}X_i - {}^{W}X_j) \times ({}^{W}X_i - {}^{W}X_j)}. $$

Here, i and j are the positions of the feet of two ground legs of a hexapod counted in the counter-clockwise sense. Thus, s_{ij} will have a positive value only when the CG of the body lies inside the support polygon which is a necessity for maintaining static stability of the vehicle.

Here, the hexapod will have to take a sharp circular turn through 90 degrees in the counterclockwise sense (taken at random) and the total distance to be traveled is divided into Q equal parts usually known as *motion segments*. The hexapod will take the decisions regarding lifting and placing of legs at the end of each motion segment.

The problem can be stated mathematically (similar to that in Chapter 6) as follows:

$$\text{Maximize} \quad z = w_1 \times (6 \times Q - C) + w_2 \times K \tag{7.3}$$

subject to the condition that the stability margin,

$$s > 0.$$

Q indicates the total number of motion segments, C is the total number of ground-legs in Q motion segments, K is the average kinematic margin of the ground-legs, s (minimum of the s_{ij} values) is the stability margin, w_1 and w_2 are the weighting factors. A penalty function approach is used here to handle the constraint of stability margin.

7.2 Description of the Algorithm

In the proposed genetic-fuzzy system, two potential tools, namely genetic algorithm (GA) (refer to Section 2.3) and fuzzy logic technique (refer to Section 2.2) have been merged to get advantages from both of them. The performance of a fuzzy logic controller (FLC) depends on its knowledge base (rule base and membership function distributions). Here, the performance of an FLC has been improved by using a GA. In this study, the rule base of an FLC has been optimized because the performance of an FLC depends more on the rule base and optimizing the membership function distribution is a fine-tuning process (Deb et al. 1998, Pratihar et al. 1998b, 1999a). The proposed genetic-fuzzy system is shown in Fig. 4.1. Here, the state of turning (to be taken) is given as input to the FLCs to generate optimal turning gait of a hexapod. In this work, the turning gait generation of a six-legged robot has been considered and the motion of each leg is controlled by a separate fuzzy logic controller. Two inputs (distance and crab angle) are given to the FLC and it produces one output (leg stroke).

The distance is the distance of the foot of a particular ground-leg from the center of rotation O (refer to Fig. 7.1). The terms crab angle and leg stroke have been defined

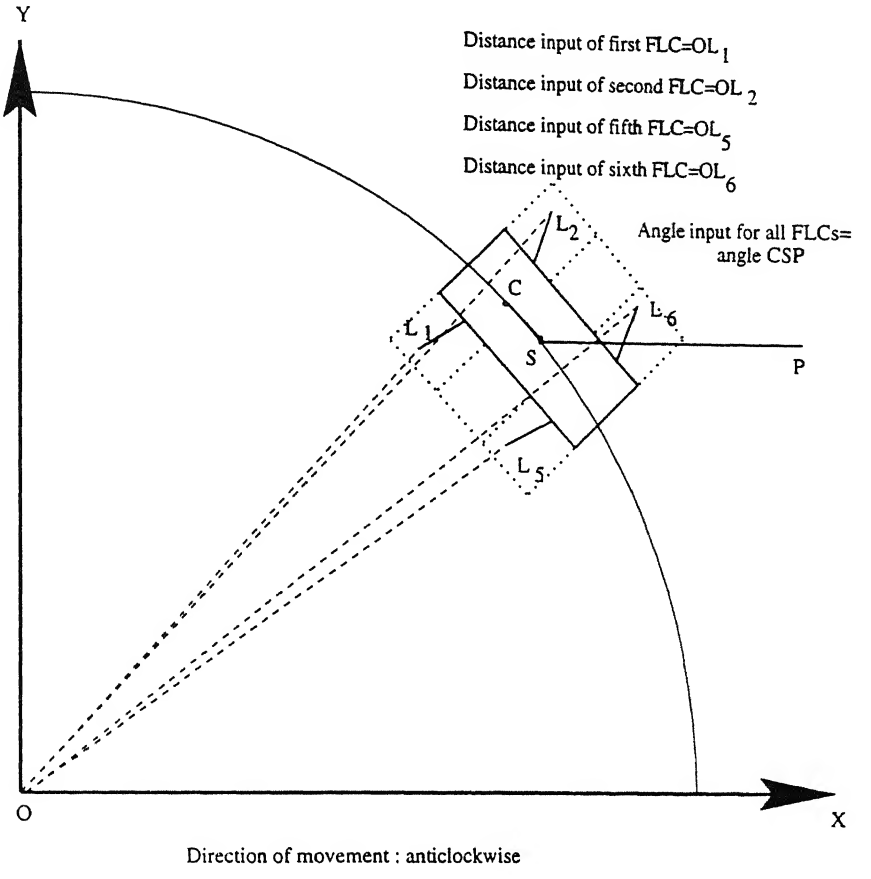


Figure 7.1: A schematic showing inputs of an FLC (turning gait).

earlier (Section 5.1.1). The *stroke control strategy* is used in the proposed algorithm. The inputs - distance and crab angle have four (VN, NR, FR, VF) and five (NL, NM, Z, PM, PL) different values, respectively. Moreover, there are six different values (VS, SM, MM, SL, LR, VL) of the output - leg stroke. Fig. 7.2 shows the author-defined membership function distributions for input and output variables used in this study. For each FLC, 20 rules are considered. Thus, there are 120 rules for all the six FLCs. Table 7.1 shows the author-defined rule base for an FLC and it is the same for all the six FLCs. Thus, a typical fuzzy rule will look as follows:

If distance is *NR* and crab angle is *NM*, then stroke is *MM*.

The following steps are considered for designing the turning gait of a hexapod:

1. The total distance to be traveled by the hexapod is divided into Q motion segments and the position of CG of the body at each motion segment is determined. In

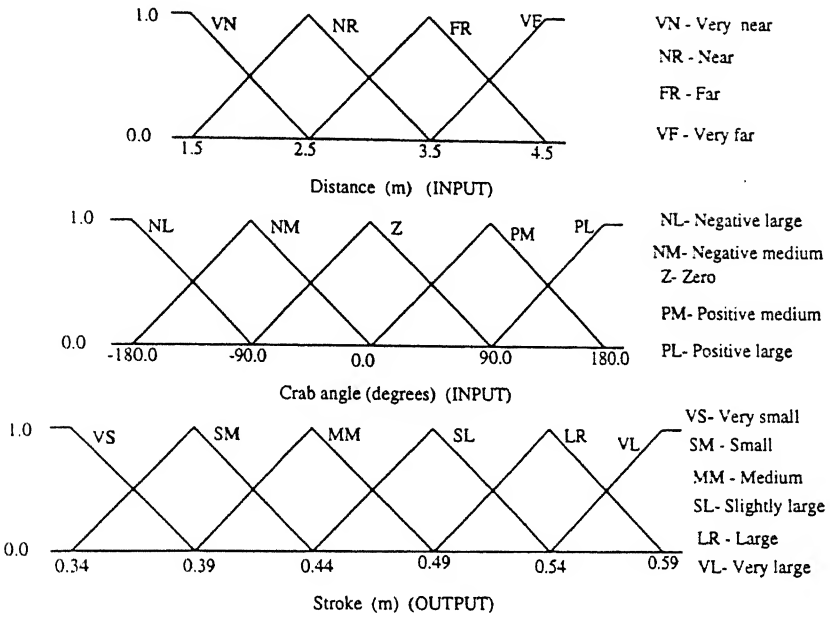


Figure 7.2: Author-defined membership function distributions for input and output of an FLC (turning gait).

turning gait generation, both translation as well as rotation of the body are to be considered.

2. The inputs (distance and crab angle) of the different fuzzy logic controllers are determined at each motion segment.

3. The stroke for each ground-leg is determined using a fuzzy logic controller.

4. The values of the kinematic margin are calculated at the immediately next and predicted support pattern (the support pattern just after transfer phase). If the

Table 7.1: Author-defined rule base for each FLC (turning gait)

distance \ crab angle	NL	NM	Z	PM	PL
VN	VS	SM	MM	SM	VS
NR	SM	MM	SL	MM	SM
FR	MM	SL	LR	SL	MM
VF	SL	LR	VL	LR	SL

kinematic margin of a particular leg is either zero or negative in the next support pattern, the leg is lifted to air.

5. At the beginning of a particular motion segment, if the kinematic margin of a leg is either zero or negative, it is selected as a transfer-leg.

6. If the stability of a vehicle is not maintained, decision regarding foot placement is taken. The leg on the air and with the highest positive kinematic margin is selected for placement on the ground to regain the stability of the body.

7. If the stability (static) is maintained, the vehicle is allowed to move.

7.2.1 Fitness Evaluation

As the variables are discrete in nature (whether a particular rule is present or absent in the rule base), a binary-coded GA (refer to Section 2.3.1) has been used here. A GA string (of length 120) will look as follows:

$$\underbrace{001\ldots1}_{\text{first FLC}}\ \underbrace{100\ldots0}_{\text{second FLC}}\ \underbrace{110\ldots0}_{\text{third FLC}}\ \underbrace{011\ldots0}_{\text{fourth FLC}}\ \underbrace{110\ldots1}_{\text{fifth FLC}}\ \underbrace{100\ldots1}_{\text{sixth FLC}}$$

Here, 1 and 0 represent the presence and absence of fuzzy rule, respectively. To find an optimum fuzzy rule base which is generic in nature, H different training scenarios (in which the turning radii are varied from one scenario to another) are considered during the optimization (tuning phase). The average z (refer to equation 7.3) is calculated and taken as actual objective function value (fitness) for the maximization problem. Since the objective is to maximize z, the GA will find its string through search which corresponds to the maximum fitness value. It is interesting to note that the GA-based tuning of fuzzy rule base is done off-line. Once the optimal fuzzy rule base is obtained, it can be used for solving the similar real-world scenarios on-line, in an optimal sense.

7.3 Results and Discussion

The following GA-parameters are selected after a careful study:

Population size = 100
Crossover probability = 0.95
Mutation probability = 0.01
Number of generations = 50

The validity of the proposed gait generation algorithm is proved by simulations (Pratihar et al. 1999d). A six-legged robot, as shown in Fig. 5.1, will have to take a sharp circular turn through 90 degrees in the anti-clockwise sense (as chosen here) and the total distance to be traveled by the hexapod is divided into $Q = 22$ (as chosen here after a careful study) motion segments. Two different approaches have been studied here, as discussed below:

Approach 1: Author-defined fuzzy logic controller. In this study, a fixed set of 120 rules and author-defined membership functions (refer to Fig. 7.2) are used. Table 7.1 shows a set of 20 author-defined rules used in one FLC and the same rule base is used in all the six FLCs. It is to be noted that the author-defined rule base and membership function distributions may not be optimal in any sense. Here, no optimization is carried out to find optimal rule base and/or optimal membership function distributions of the FLCs.

Approach 2: Optimizing rule base alone of the FLCs. Here, the rule base of the FLCs is optimized keeping the membership function distributions same as shown in Fig. 7.2, using a GA. As the maximum number of possible rules is 120, a binary-coded GA with 120-bit string is used in which 1 and 0 indicate presence and absence of rules, respectively. A GA will find through search the optimal rule base that will result in a situation in which the hexapod will take a sharp circular turn in the anti-clockwise direction with minimum number of ground-legs having the maximum average kinematic margin and after satisfying the constraint of stability margin. In this study, H=10 (taken at random) different author-defined scenarios (having different turning radii) are used in the training phase. The weighting factors w_1 and w_2 are finally set to 1.0 and 9.0, respectively, after a careful study with different sets of values.

The minimum number of ground-legs, C and the maximum average kinematic margin of the ground-legs, K obtained during the turning gait generation are presented in Table 7.2 for the two approaches mentioned above. In this table, the first three rows show the

Table 7.2: Number of ground-legs, C and average kinematic margin of ground-legs, K obtained by two approaches (turning gait)

Scenario	Approach 1		Approach 2	
	C	K	C	K
1	95	1.54	90	1.63
2	95	1.56	93	1.63
3	95	1.56	94	1.62
4	99	1.55	94	1.63
5	95	1.56	94	1.63
6	94	1.55	94	1.56
7	-	-	87	1.57

three training scenarios (out of 10) used during the tuning phase. The subsequent four rows show four different and new scenarios which were not used during the optimization process. In scenario 7 (Table 7.2), it is observed that the author-defined FLCs have failed to generate stable gaits for the hexapod, whereas the GA-designed FLCs have successfully generated the stable gaits although this case was not used during the optimization phase. In all cases, the GA-designed FLCs have performed better than the author-defined ordinary FLCs. It happens due to the fact that the knowledge base of the author-defined FLCs may not be optimal in any sense. The optimized rule base obtained through Approach 2 for first through sixth FLCs are shown in Tables 7.3 through 7.8, respectively. It is observed that only 65 rules (out of 120 author-defined rules) are selected by the GA for all the six FLCs. It is intuitive to say that there are still some redundant rules in the obtained rule base and a second stage GA- based tuning will further reduce the number of good rules for all the six FLCs. It is important to note that the presence of rules in the optimum rule base depends on the nature of training cases considered. Fig. 7.3 shows the path traced by the CG of the hexapod while taking the turn through 90 degrees in the anti-clockwise direction. The generated gaits obtained using Approach 2 for the test scenario 4 (refer to Table 7.2) are shown in Fig. 7.4. It is observed from Fig. 7.4 that, in most cases, the inner legs are placed more away from the body center than the outer legs. This is a strategy which is necessary to maintain stability while turning. Although

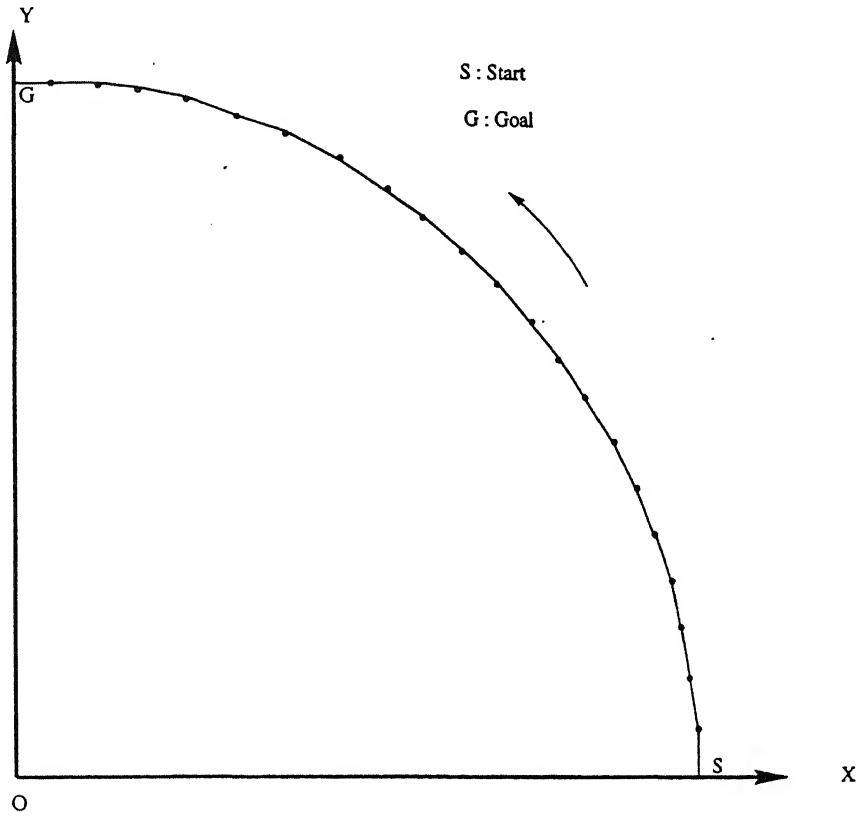


Figure 7.3: Path traced by the CG of the vehicle (turning gait).

this information was not explicitly given to the FLCs, the genetic-fuzzy system finds this information through search.

Table 7.3: Optimized rule base for first FLC (having thirteen rules only) obtained using Approach 2 (turning gait)

distance \ crab angle	NL	NM	Z	PM	PL
VN	VS		MM		VS
NR	SM		SL	MM	SM
FR	MM			SL	MM
VF	SL	LR	VL		

Table 7.4: Optimized rule base for second FLC (having eight rules only) obtained using Approach 2 (turning gait)

distance \ crab angle	NL	NM	Z	PM	PL
VN	VS	SM			VS
NR	SM	MM			
FR	MM	SL			
VF			VL		

Table 7.5: Optimized rule base for third FLC (having eleven rules only) obtained using Approach 2 (turning gait)

distance \ crab angle	NL	NM	Z	PM	PL
VN	VS			SM	
NR	SM			MM	SM
FR	MM	SL	LR	SL	MM
VF				LR	

Table 7.6: Optimized rule base for fourth FLC (having ten rules only) obtained using Approach 2 (turning gait)

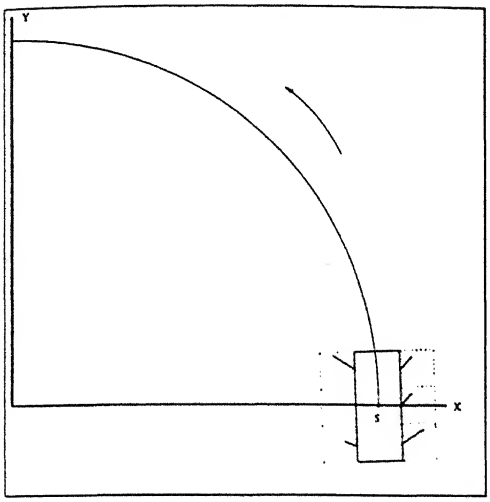
distance \ crab angle	NL	NM	Z	PM	PL
VN	VS	SM	MM		
NR	SM				SM
FR	MM		LR		
VF		LR		LR	SL

Table 7.7: Optimized rule base for fifth FLC (having nine rules only) obtained using Approach 2 (turning gait)

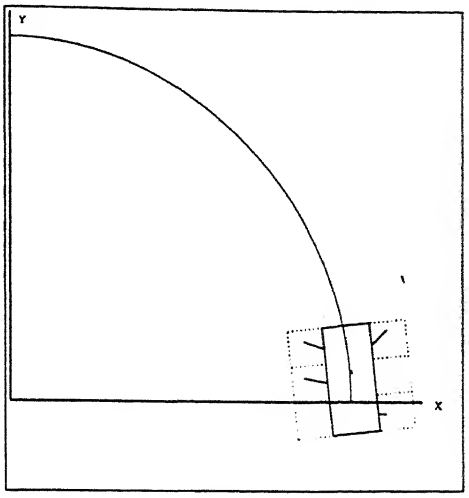
distance \ crab angle	NL	NM	Z	PM	PL
VN	VS	SM			
NR	SM			MM	SM
FR				SL	MM
VF			VL	LR	

Table 7.8: Optimized rule base for sixth FLC (having fourteen rules only) obtained using Approach 2 (turning gait)

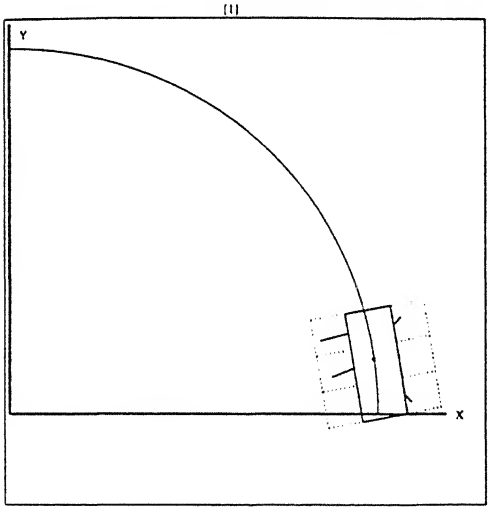
distance \ crab angle	NL	NM	Z	PM	PL
VN	VS	SM	MM		
NR	SM	MM	SL		SM
FR	MM			SL	MM
VF	SL	LR	VL	LR	



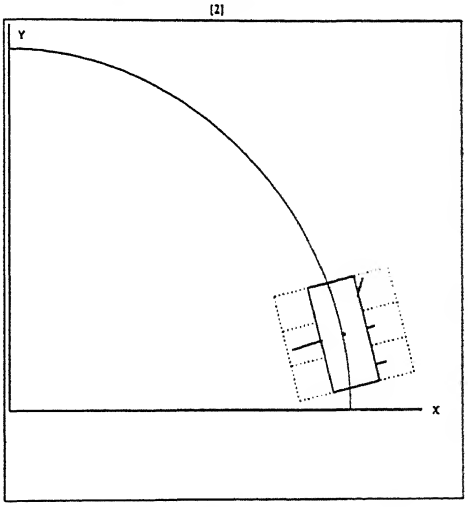
[1]



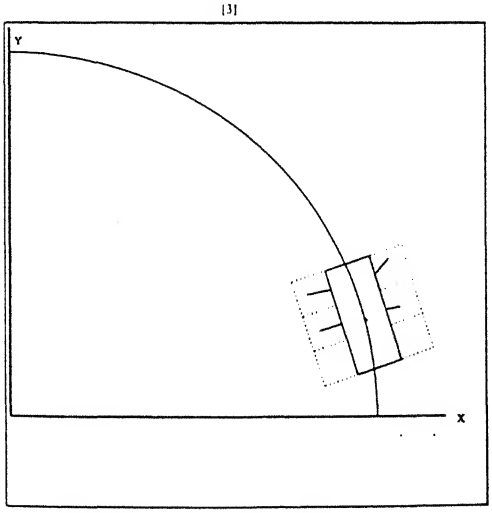
[2]



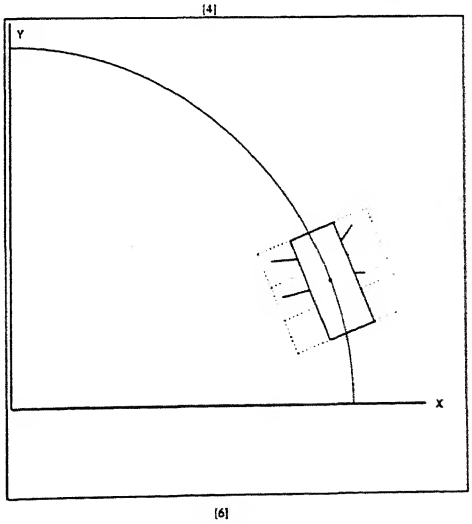
[3]



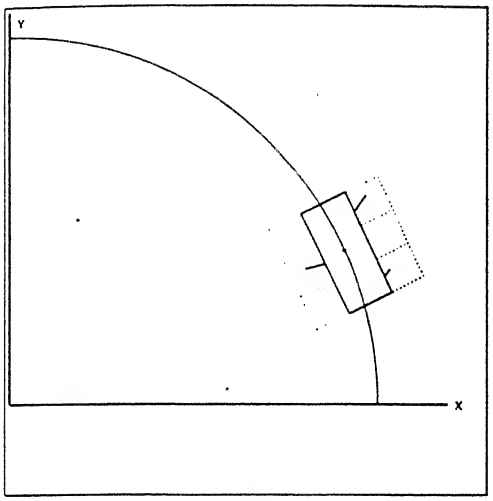
[4]



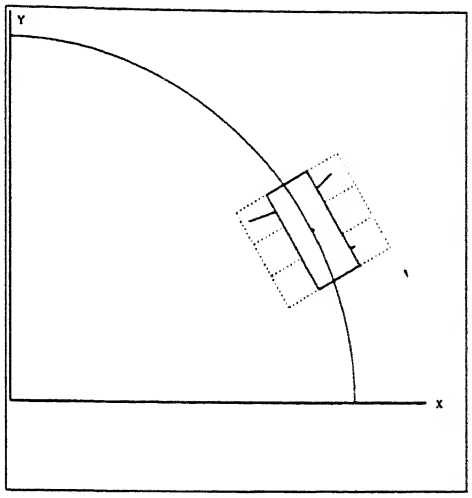
[5]



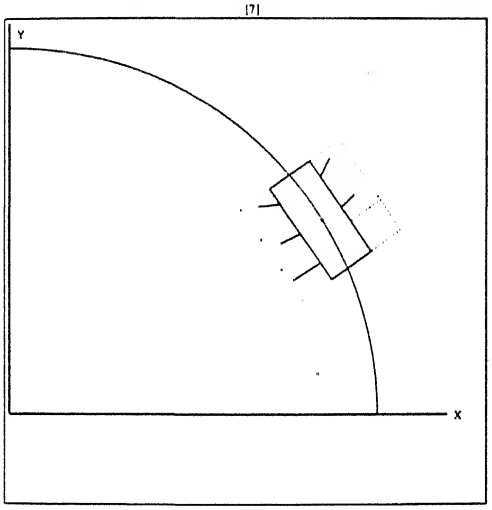
[6]



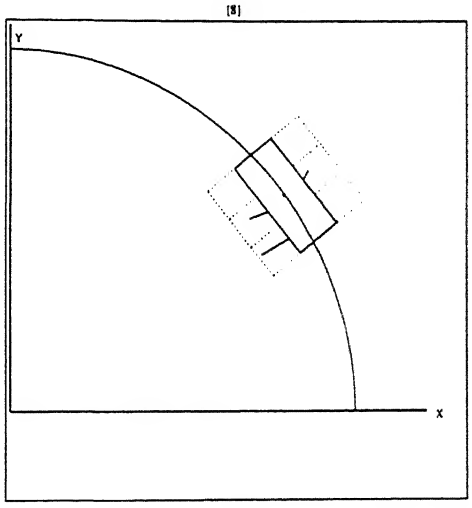
[7]



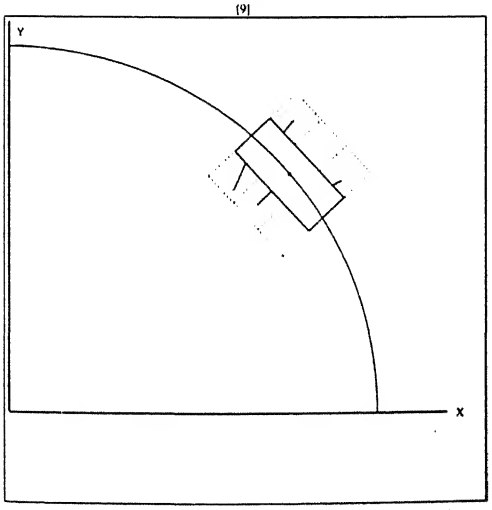
[8]



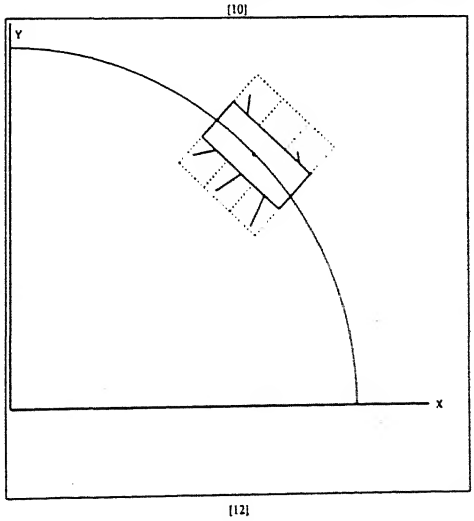
[9]



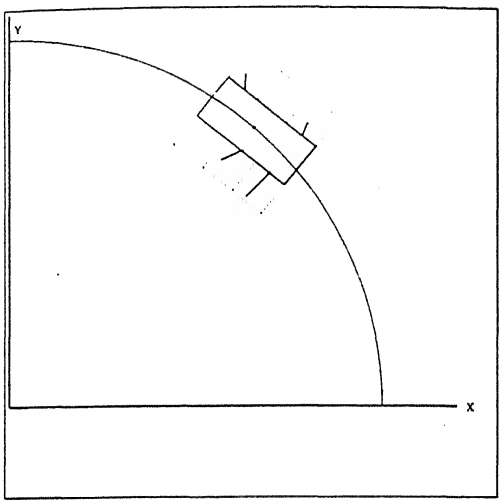
[10]



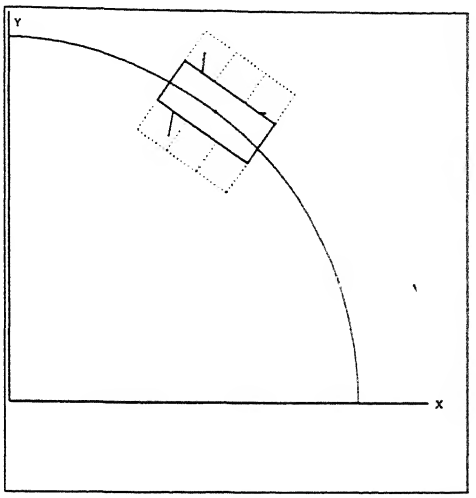
[11]



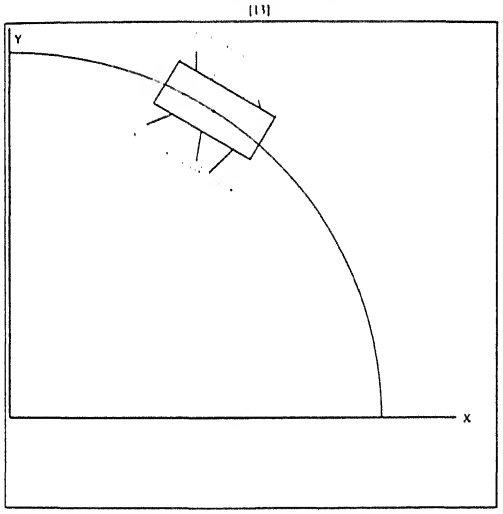
[12]



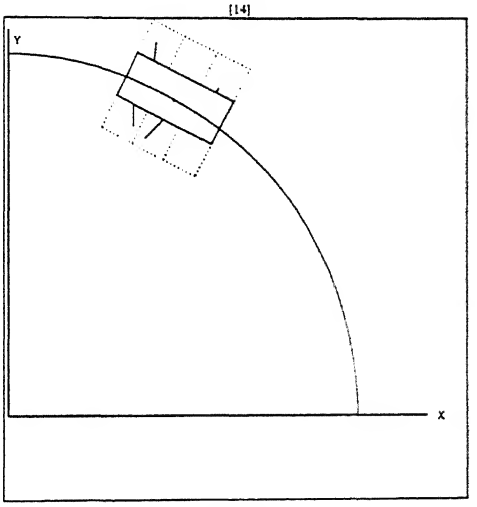
[13]



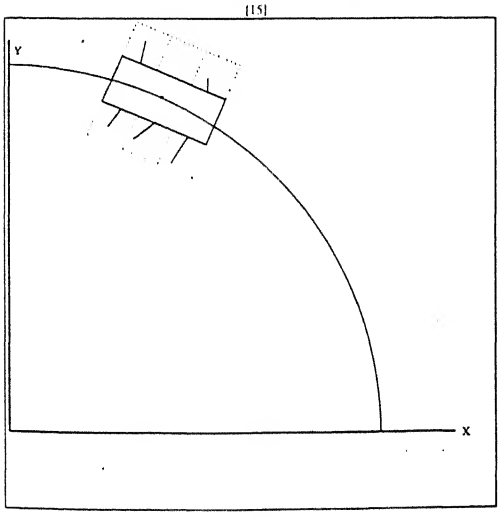
[14]



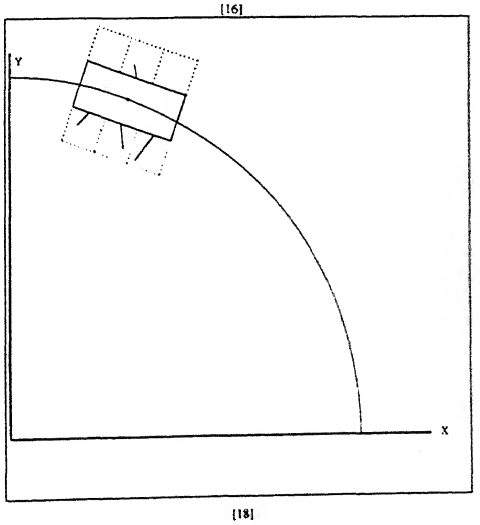
[15]



[16]



[17]



[18]

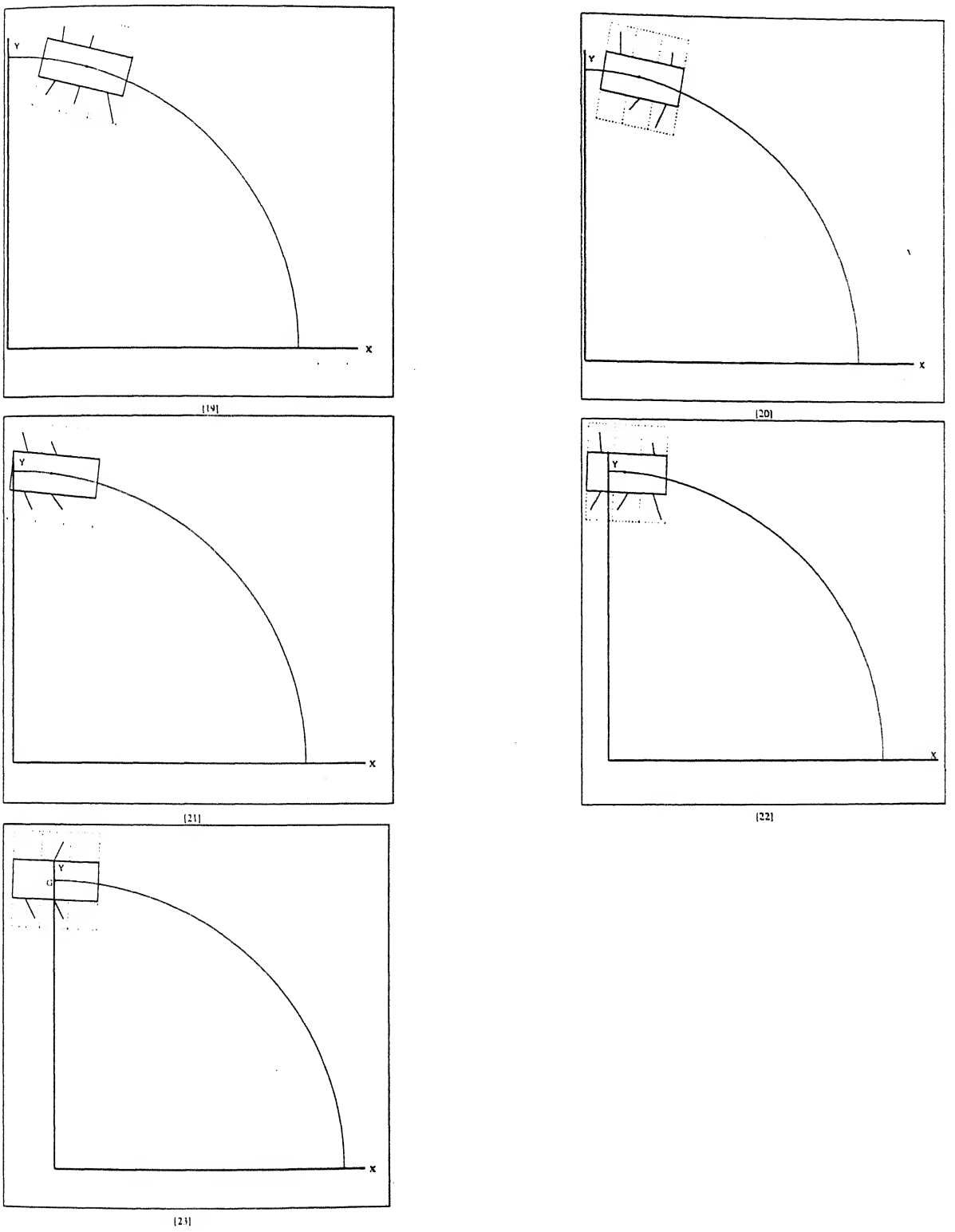


Figure 7.4: Generated gaits obtained using Approach 2 for test scenario 4 (Table 7.2)

7.3.1 Comparison of the Proposed Algorithm with the Graph Search Technique

Pal et al. (1994) proposed the graph search technique in which the support state of the legs was expressed as a binary number (1 if the leg is on the ground and 0 if it is in the air) to solve the turning gait generation problems of a legged robot. For generating the optimal gait, a support state transition table was formed based on the criterion of optimization and the A* algorithm was used to find an optimal route to the goal. This approach is not suitable for on-line implementation due to its computational complexity, whereas the proposed algorithm can be implemented on-line, due to its low computational load (the execution time is found to be less than 1.0 sec in a HP 9000/K200 machine). Moreover, the solutions of the graph search technique may be locally optimal. As GA is a population-based search technique, the chance of its solution for getting trapped into local minima is less.

7.4 Summary

A genetic-fuzzy system has been used, in this chapter, to generate optimal/ near-optimal turning gaits of a six-legged robot. In this approach, a GA is used to design the optimized FLCs. As the variables are discrete in nature (rule base optimization), a binary-coded GA is a natural choice for solving this type of problems. There are six FLCs running in parallel to control the six legs of the robot. The GA-tuned FLCs are found to perform better than the author-defined FLCs. It is interesting to note that the proposed algorithm overcomes the limitations of the graph search technique proposed by Pal et al. (1994). The graph search technique is not suitable for on-line implementation due to its high computational load, whereas the proposed genetic-fuzzy system can be implemented on-line. It is to be noted that after the training (which is to be done off-line) is over, the proposed algorithm is suitable for on-line implementation due to its low computational load. Moreover, the solution of the graph search technique along with A* algorithm may be locally optimal. On the other hand, since GA is a population-based search and optimization technique, the chance of its solution for getting trapped into local minima is less.

Chapter 8

COMBINED PATH AND GAIT GENERATIONS OF A LEGGED ROBOT

A genetic-fuzzy approach is used, in this chapter, to solve the problem of combined path and gait generation of a legged robot. A six-legged robot will have to determine its time-optimal, collision-free path as well as optimal gait (with minimum number of ground-legs having the maximum average kinematic margin). It is a complicated task because the path planning and gait planning are to be done simultaneously. A GA is used to design optimized FLCs off-line and seven (one is for path planning and the remaining six are for gait planning) such optimized FLCs are running, in parallel, to generate optimal path and gait simultaneously of a six-legged robot.

8.1 Problem Formulation

A six-legged robot as shown in Fig. 5.1, will have to plan its path as well as gait simultaneously, while moving on flat terrain with occasional hurdles such as ditches and in presence of moving obstacles. The three steps to be followed for legged robot locomotion are as follows:

1. Determination of vehicle's trajectory,

2. Foothold selection, and

3. Design of a sequence of leg movements.

In practice, path and gait generations are to be done simultaneously. The hexapod needs to do this job by avoiding to collide with any moving obstacles and not falling into ditches and all within the minimum time of travel and with an optimum effort-to-gain ratio (with minimum number of legs on the ground and with maximum average kinematic margin (Section 5.1.1) of the ground-legs). Moreover, its stability margin (Section 5.1.1) should always be positive to ensure static stability. There exists a couple of other issues: (i) in such problems, as the number of ground-leg increases, the probability of a deadlock situation (Section 5.1.1) occurring increases, (ii) as the average kinematic margin of the ground-legs is more, the potential progress of the vehicle towards the goal is also more. To perform the above tasks, in practice, the hexapod will have to do the following three sub-tasks optimally (with minimum travel time):

1. Move along straight-line paths (only translation),

2. Take sharp circular turns (translation and rotation),

3. Cross ditches (translation).

It is clear that the tasks are complex and the following assumptions are made to simplify the problem:

1. Based on the available sensory information, the terrain is discretized into a number of cells and the center of each cell is considered as a candidate foothold.

2. There exists a point contact (with no slipping) between the foot of a leg and the ground.

3. The mass of all the legs is lumped into the body and the center of gravity is assumed to be at the centroid of the body.

4. Each obstacle is represented by its bounding circle.

The vehicle's complete path (from point S to point G) is a collection of a number of small straight-line segments and circular arcs each traveled for a constant time step, ΔT. It is assumed that the hexapod starts from zero velocity and accelerates during the time $\Delta T/4$ and then maintains a constant velocity $a\Delta T/4$, where a is the acceleration. The magnitude of acceleration is assumed to be equal to that of deceleration. The vehicle senses the position of each moving obstacle just before the end of time step, ΔT and decides whether to continue moving in the same direction or to change its direction of movement. This is achieved by first determining the predicted position of each obstacle, as follows:

$$P_{\text{predicted}} = P_{\text{present}} + (P_{\text{present}} - P_{\text{previous}}). \tag{8.1}$$

If the robot has to change its direction of movement, its velocity is reduced to zero at the end of the time step; otherwise the robot does not decelerate and it will continue moving in the same direction with the same velocity $a\Delta T/4$. It is to be noted that when the latter case occurs (the robot does not change its direction of movement) in two consecutive time steps, there is a saving of $\Delta T/4$ sec in travel time per such occasion. The robot will start decelerating whenever it comes closer to the destination (point G) so that it can reach point G with a zero velocity. Fig. 3.2 shows the velocity and acceleration distributions of the vehicle along the trajectory. The total travel time, T is then calculated as follows:

$$T = \frac{D}{v} + \frac{v}{a}, \tag{8.2}$$

where, D is the total distance traveled, v is the maximum velocity of the vehicle along the trajectory.

It is important to note that there can be several feasible solutions of a motion planning problem but all of them may not be optimal in terms of traveling time. A fixed value of maximum traveling time, T_{max} has been assumed. If the true traveling time T (to reach final point G) is less than T_{max}, the true time T is used as objective function value for the solution. On the other hand, if the actual traveling time, T exceeds T_{max} but the robot does not reach its destination, it is halted at its current position and the Eucleadian distance d_{rem} between the halted position and the destination point is calculated. The objective function value for this case is modified approximately as $T = T_{max} + d_{rem}/v$.

Besides time-optimal path planning, the hexapod will have to determine its optimal gait. While navigating from an initial position to a destination, the hexapod will have to move along a straight path (periodic gait), to take a circular turn (non-periodic gait),

to cross a ditch (non-periodic gait) as the situation demands. For each mode of gait generation, the distance to be traveled by the vehicle is divided into Q *motion segments*. The following steps are considered for designing the gait of a hexapod.

1. Determine the position of CG of the vehicle at each motion segment.
2. Calculate the values of condition variables (inputs) for the FLCs. The stroke for each ground-leg is determined using a fuzzy logic controller.
3. Decision regarding the lifting of leg to the air, is taken based on the kinematic margin calculation at the immediately next and predicted support pattern (the support pattern just after transfer phase). If the kinematic margin of a particular leg is either zero or negative in the next support pattern, the leg is lifted to the air.
4. The leg to be in transfer phase, is decided at the beginning of a particular motion segment. The leg with a zero or negative kinematic margin is selected as a transfer-leg.
5. If the stability (static) is maintained, the vehicle is allowed to move. Otherwise, decision regarding foot placement is taken. The leg on the air and with the highest positive kinematic margin is selected for placement on the ground to regain its stability. The CG should always lie inside the support polygon to ensure static stability of the vehicle.

Periodic gait (wave gait) is optimal in terms of ground-leg. Thus, no optimization is carried out in periodic gait generation mode and an attempt will be made to optimize the non-periodic gaits only. In optimal gait generation mode, the hexapod will place minimum number of legs on the ground with a maximum average kinematic margin of the ground-legs.

Two reference frames, namely world coordinate frame $\{W\}$ and body coordinate frame $\{B\}$ have been considered and the expression for the (stability) margin s_{ij} can be derived in the similar way, as done in Sections 5.2, 6.1 and 7.1. The problem can be stated mathematically as follows:

$$\text{Maximize} \quad z = \sum_{i=1}^{NPG} (w_1 \times (6 \times Q_i - C_i) + w_2 \times K_i) \tag{8.3}$$

subject to the condition that the stability margin is restricted as

$$s > 0,$$

where, NPG indicates the total number of non-periodic gait generation mode, Q_i is the number of motion segments in the i-th mode, C_i indicates the total number of ground-legs in the i-th mode, K_i is the average kinematic margin of the ground-legs in the i-th mode, w_1 and w_2 are the weighting factors and s (minimum of the s_{ij} values) is the stability margin of the vehicle. It is important to mention that a penalty term is included in the objective function if the constraint of stability margin is violated.

8.2 Description of the Proposed Algorithm

Fig. 4.1 shows the schematic diagram of a *genetic-fuzzy* system used to solve the problem of combined path and gait generations of a hexapod. The state of obstacles, ditch and turning is considered as input to the FLCs to generate an obstacle-free, time-optimal path and optimal gaits of a six-legged robot. Here, a binary-coded GA (Section 2.3.1) has been used as an optimization tool to improve the performance of an FLC which depends on its rule base as well as membership function distribution. It has been observed that optimizing rule base of an FLC is a rough tuning process, whereas optimizing membership function distribution is a fine tuning process (Deb et al. 1998, Pratihar et al. 1998b, 1999a). Thus, in this study, the rule base alone is optimized keeping the membership function distribution unaltered (to keep the matters simple and yet to show the efficacy of the proposed approach). Here, a GA will try to find those rules from an author-defined large rule base so that the performance is optimized. Here, there is one FLC for determining the collision-free, time-optimal path of the vehicle and each leg of the hexapod is controlled by a separate FLC. Thus, there are seven (one for path generation and six for gait generation) FLCs, running in parallel, to solve the problem of combined path and gait generations. At the beginning of each time step, ΔT, the vehicle will first search whether there is any ditch ahead. If it finds a ditch, the ditch crossing module will be activated, otherwise, it will move along a straight path following the periodic gait pattern during the time step ΔT. After moving along a straight path or crossing a ditch, the vehicle will take a sharp circular turn during the next time step ΔT if there is any change in direction of its movement. It is obvious that the vehicle will continue moving

in the same direction during the next time step ΔT if there is no change in its direction of movement. This process will continue until it reaches the destination. Fig. 8.1 shows the flowchart of the proposed algorithm. Thus, the combined problem (of path and gait

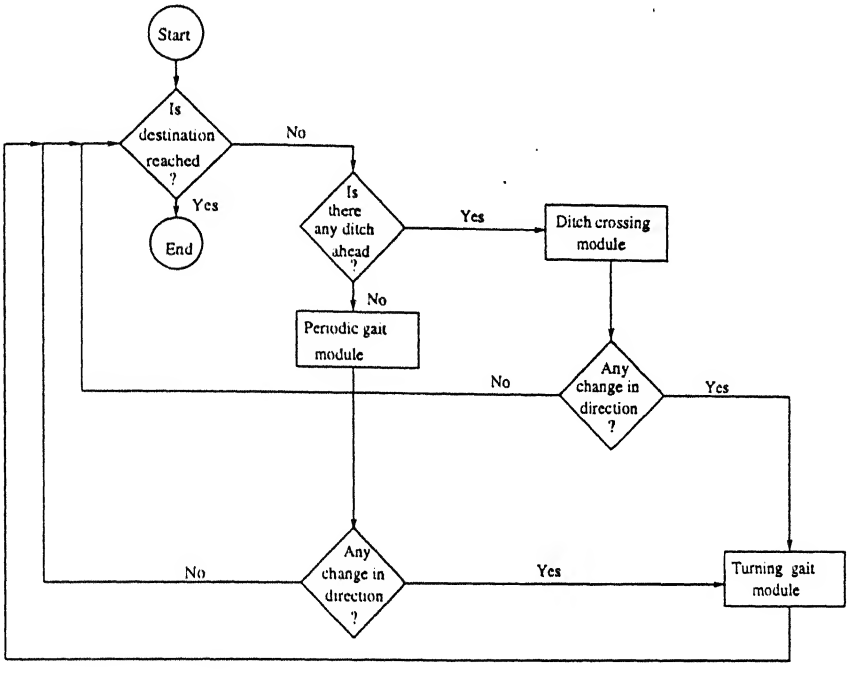


Figure 8.1: Flowchart of the proposed algorithm - combined path and gait generation.

generations) includes many modules, namely the path generation module, periodic gait generation module, ditch crossing module and turning gait generation module. All these modules have been discussed here, in detail.

8.2.1 Path Generation Module

A time-optimal collision-free path of a mobile robot is determined by using an FLC whose performance is tuned by a GA. The FLC finds the obstacle-free direction based on the predicted position of the obstacles in the next time step. It has been discussed, in detail, in the Chapter 4. Two inputs, namely distance and angle are fed to the fuzzy logic controller and there is one output of the FLC, that is, deviation. The condition (distance and angle) and action (deviation) variables of the FLC are shown in Fig. 4.2. It is important to note that relative velocity also plays an important role in determining the critical obstacle for the robot. Four different values of distance, namely very near (VN), near (NR), far (FR)

and very far (VF) are considered here. Similarly, five different fuzzy values are assigned for both angle and deviation, namely left (LT), ahead left (AL), ahead (A), ahead right (AR), and right (RT). Fig. 8.2 shows the membership function distributions for input and output variables. As there are four and five different values for distance and angle,

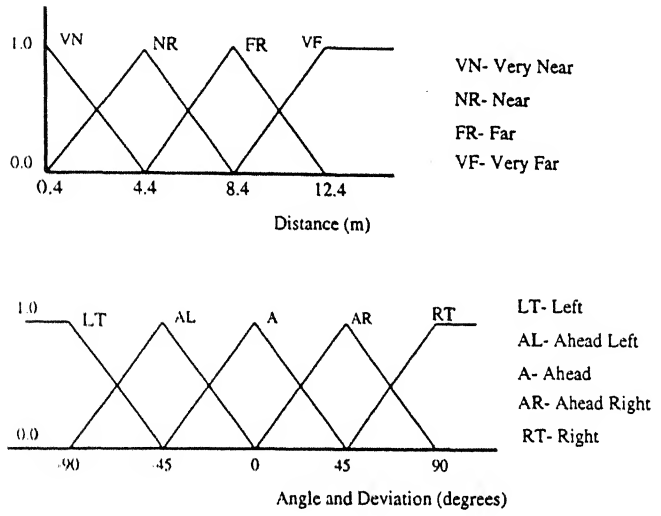


Figure 8.2: Author-defined membership function distributions for input and output of an FLC - path generation module.

respectively, 20 rules are considered in the manually constructed rule base of the FLC. All possible rules of the author-defined rule base are shown in Table 4.1. A GA will try to find the good rules through search from this author-defined large rule base so that the performance of an FLC is optimized.

8.2.2 Periodic Gait Generation Module

Periodic gaits are generated by a legged robot only when it moves on a smooth terrain, along a straight path. For a periodic gait (wave gait) with a duty factor β (refer to Section 5.1.1), the phases of legs 1,3,5,2,4,6 with respect to leg 1 are 0, β, $2\beta - 1$, $1/2$, $\beta - 1/2$, $2\beta - 1/2$, respectively (Pal et al. 1994). The sequence of leg transfer is determined for fixed values of duty factor, $\beta(= 2/3)$ and phase difference, $\gamma(= 1/2)$, as shown in Fig. 5.4. As the wave gait is optimal in terms of ground-legs, there is no scope of further optimization. All the six FLCs used for controlling the legs have the same and fixed output, that is, stroke. Here, the fixed leg-stroke is assumed to be equal to the distance

traveled by the vehicle (CG of the body) during each motion segment. Chapter 5 provides a detailed discussion on periodic gait generation.

8.2.3 Ditch Crossing Module

There are two inputs (distance and relative angle) and one output (leg stroke) of the fuzzy logic controller. The inputs of an FLC are shown in Fig. 6.1. The proposed algorithm is based on the *stroke control strategy*. Four (VN, NR, FR, VF) and five (NL, NM, Z, PM, PL) different values are considered for distance and relative angle, respectively. Moreover, the output stroke has got six (VS, SM, MM, SL, LR, VL) different values. The membership function distributions for input and output used in this study are shown in Fig. 6.2. For each FLC, 20 rules are considered. Thus, 120 rules are considered for all the six FLCs. The author-defined rule base of an FLC is shown in Table 6.1 and it is same for all the six FLCs. Here, a GA will select a set of good rules from this manually constructed large rule base so that the hexapod can cross the ditch in an optimal sense. It has been discussed, in detail, in the Chapter 6.

8.2.4 Turning Gait Generation Module

Two inputs (distance and crab angle) are fed to the FLC and it produces one output (stroke). The inputs of an FLC are shown in Fig. 7.1 in which the hexapod is taking a sharp circular turn through 90 degrees in the counter-clockwise sense (chosen arbitrarily). It is important to note that the hexapod will cover this circular path in one time step, ΔT which is divided into Q motion segments. The *stroke control strategy* is used in the proposed algorithm. There are four (VN, NR, FR, VF) and five (NL, NM, Z, PM, PL) different values for distance and crab angle, respectively. Moreover, six (VS, SM, MM, SL, LR, VL) different values are considered for the output - (stroke). Fig. 8.3 shows the author-defined membership function distributions for input and output of the FLCs used in this study. As there are four and five different values for distance and crab angle, respectively, 20 rules are considered for each FLC. Thus, a set of 120 rules are considered for all the six FLCs. Table 7.1 shows the author-defined rule base for an FLC and the same rule base is used for all the six FLCs. A GA will find only those rules from this large rule base which will optimize the performance of the FLCs. It has been studied, in

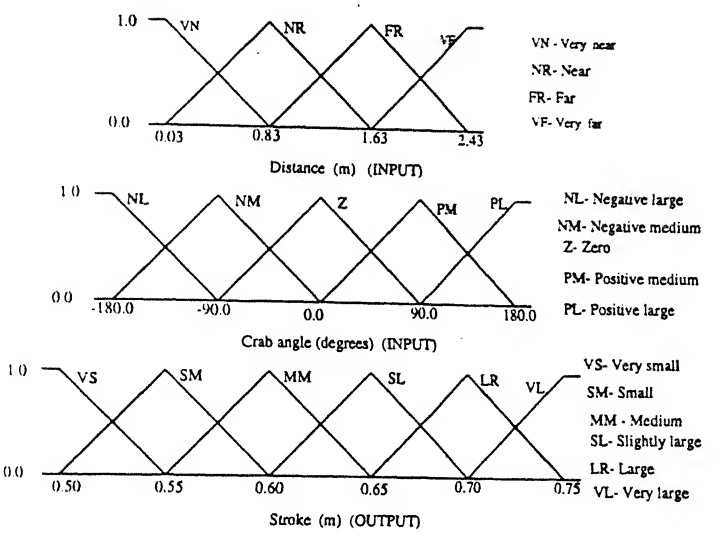


Figure 8.3: Author-defined membership function distributions for input and output of the FLCs - turning gait generation module.

detail, in the Chapter 7.

8.2.5 GA Representation of a Solution

Here, a binary-coded GA with 260-bit string is used to represent a solution. The GA string will look as follows:

$$\underbrace{10111\ldots1}_{\text{path}}\underbrace{01100\ldots01100111}_{\text{ditch crossing gait}}\underbrace{11001\ldots10001101}_{\text{turning gait}}$$

Here, 1 and 0 represent the presence and absence of fuzzy rule, respectively. The first 20-bits in this string will represent the information regarding path generation, the next 120-bits will give gait generation information in the ditch crossing mode and the remaining 120-bits will carry information regarding turning gait generation.

8.2.6 Fitness Evaluation

A GA is used to find optimal or near-optimal path and gaits simultaneously for the hexapod. Here, a binary-coded GA is used which begins its search by randomly creating a number of solutions. Each solution in this problem is represented in a 260-bits string

(20-bits for path generation and remaining 240-bits for gait generation). The fitness value for each solution is evaluated as follows:

The hexapod will have to reach its destination starting from an initial position in minimum traveling time and its generated gaits will be optimal (with minimum number of ground-legs and with maximum average kinematic margin of the ground-legs). All these criteria are considered in a combined form (combining equations 8.2 and 8.3) of objective function, as given below:

$$\text{Maximize} \quad f = \eta_1 \frac{1}{T} + \eta_2 z, \tag{8.4}$$

where, η_1 and η_2 are two weighting factors. Since two weighting factors (w_1 and w_2) have already been used in the z term (equation 8.3), these two weighting factors (η_1 and η_2) are not explicitly used here. Instead, both η_1 and η_2 are set to 1, here. It is important to note that this problem may be posed as a multi-objective optimization problem and multi-objective GA implementations can be used to find multiple Pareto-optimal solutions (Srinivas and Deb 1995). But, here, the above mentioned weighting scheme has been used to solve it as a single-objective optimization problem.

Each solution is then evaluated to calculate a function value f_j for H different training scenarios ($j = 1, 2 \ldots, H$). The fitness of the string is assigned as $FS = \left(\sum_{j=1}^{H} f_j\right) / H$. After each solution in the population is evaluated and fitness is assigned, the population is modified by using three GA-operators, namely reproduction, crossover and mutation (refer to Section 2.3.1).

8.3 Results and Discussion

The performance of a GA depends on the selection of its different parameters, namely population size, crossover probability, mutation probability etc. The best performance is observed with the following parameters:

Population size = 100
Crossover probability = 0.95
Mutation probability = 0.02
Number of generations = 50

The effectiveness of the proposed algorithm is tested through simulations (Pratihar et al. 1999e, f). A six-legged robot (rectangular body plate with sides 0.9×0.4 m^2) will have to plan its path and gaits simultaneously, in optimal sense. In this study, the distance step is divided into $Q = 15$ motion segments. The acceleration and deceleration, a is assumed to be equal to 0.333 m/sec^2. It is also assumed that the vehicle will accelerate and decelerate during the first 3 sec and the last 3 sec of its travel, respectively. Thus, the maximum velocity of the robot comes out to be 1 m/sec. After a careful study, the weighting factors - w_1 and w_2 are set to 1.0 and 6.0, respectively (refer to equation 8.3). Ten different training scenarios are considered during the tuning phase.

Two different approaches have been studied, here, as discussed below:

Approach 1: Author-defined fuzzy logic controller. In this approach, a fixed set of 260 rules (20 rules for generating path and 240 rules for generating gaits) and author-defined membership functions (refer to Figs. 8.2, 6.2 and 8.3) are used. Table 4.1 shows a set of 20 author-defined rules of an FLC used for path generation. Table 6.1 shows a set of 20 author-defined rules of an FLC used in the ditch crossing module and the same rule base is used in all the six FLCs. Thus, there is a fixed set of 120 rules in the ditch crossing module. Similarly, a set of 20 author-defined fuzzy rules (used in the turning gait generation module) are shown in Table 7.1 and the same rule base is used in all the six FLCs. Thus, there is a fixed set of 120 rules for generating the turning gait. It is to be noted that the author-defined rule base and membership function distributions may not be optimum. Here, no optimization is carried out to find optimal rule base of the FLCs.

Approach 2: Tuning rule base alone of the FLCs. In this study, the rule base of the FLCs has been optimized keeping the membership function distributions same as shown in Figs. 8.2, 6.2 and 8.3. The maximum number of possible rules is 260. Here, a binary-coded GA with 260-bit string is used in which 1 and 0 indicate presence and absence of rules, respectively. Thus, a GA will find through search a set of good rules from these 260 rules so that the hexapod will be able to plan its path and gaits simultaneously, in the optimal sense (with minimum traveling time, with minimum number of ground-legs and with maximum average kinematic margin of the ground-legs) after satisfying the constraint of stability margin. Here, H=10 (chosen arbitrarily) different author-defined training scenarios (in which the size and location of the obstacles and the ditch are varied) are considered in the

optimization phase.

The problems of combined path and gait generations are solved using both the approaches mentioned above. The minimum number of ground-legs, C, the maximum average kinematic margin of the ground-legs, K, traveling distance, D, and time, T are presented for the two approaches in Table 8.1. In this table, the first three rows show the three

Table 8.1: Number of ground-legs, C, average kinematic margin of ground-legs, K, traveling distance, D (m) and time, T (sec) obtained by two approaches (combined path and gait generation)

Scenario	Approach 1				Approach 2			
	C	K	D	T	C	K	D	T
1	332	1.32	37.28	40.28	329	1.36	37.17	40.17
2	544	1.37	42.53	45.53	325	1.33	37.07	40.07
3	336	1.29	37.40	40.40	329	1.35	37.16	40.16
4	547	1.41	42.90	45.90	328	1.35	37.15	40.15
5	546	1.39	42.57	45.57	329	1.37	37.20	40.20
6	-	-	-	-	327	1.33	37.12	40.12

(out of 10) training scenarios. The subsequent three rows show three different and new scenarios which were not used during the optimization process. In scenario 6 (Table 8.1), it is seen that the author-defined FLCs have failed to generate stable gaits for the hexapod, whereas the GA-tuned FLCs have successfully done it although this case was not considered during the optimization phase. In all cases, the GA-tuned FLCs are found to perform better than the author-defined FLCs. It is due to the fact that the knowledge base of the author-defined FLCs may be far from being optimal. Table 8.2 shows the GA-tuned rule base of an FLC used for generating the time-optimal, collision-free path of the hexapod. It is interesting to note that obstacles 1 and 2 are approaching the robot from its left side, whereas obstacle 3 is approaching the robot from the right side. This fact is reflected on the optimized rule base shown in Table 8.2. There are three rules when the input - angle is LT and three other rules when the input - angle is AR and RT (refer to Table 8.2). The optimized rule bases for first through sixth FLCs used in the ditch crossing module are shown in Tables 8.3 through 8.8, respectively. It is to be noted that there are comparatively more number of rules present in the GA-tuned rule base particularly when the input - relative angle is NM, this happens due to the nature of the

Table 8.2: Optimized rule base (having six rules) for an FLC - path generation module

distance \ angle	LT	AL	A	AR	RT
VN				AL	A
NR	A			A	
FR	A				
VF	A				

training cases considered. It is also intuitive to say that there are still some redundant rules in the optimized rule base and a second stage GA-based tuning will further reduce the number of rules to be present in the optimized rule base. Similarly, Tables 8.9 through 8.14 show the optimized rule bases obtained for first through sixth FLCs, respectively, in the turning gait generation module. In this module, it is assumed that the robot can take a sharp circular turn in either clockwise or anti-clockwise sense. For a particular training case, the robot may take some turns in the clockwise and some other turns in the anti-clockwise directions. Thus, the two inputs, namely distance and crab angle are varying continuously, in their respective ranges. It is also intuitively expected that the GA-tuned rule bases will contain more number of rules (widely distributed). Thus, a GA has selected only 115 rules (6 for path generation and 109 for gait generations) out of 260 author-defined rules. Fig. 8.4 shows the collision-free paths generated by the robot using both the Approaches 1 and 2. Fig. 8.5 shows the generated path and gaits obtained using Approach 1 for test scenario 4 (Table 8.1) at the 10-th, 20-th, 35-th, 50-th, 65-th and 79-th motion segment counted from the initial position of the robot. Similarly, the generated path and gaits obtained using Approach 2 for the test scenario 4 (Table 8.1) at the 10-th, 20-th, 35-th, 50-th, 65-th and 79-th motion segment are shown in Fig. 8.6. It is interesting to note that in Approach 2, the vehicle is able to reach its destination only at 79-th motion segment starting from its initial position, whereas in Approach 1, the vehicle is almost on its mid-way at the same 79-th motion segment. It is important to note that the execution time of the proposed algorithm is found to be 1.0 sec in a HP-9000/K200 machine. Thus, it can be implemented on line, for determining optimal path and gaits simultaneously of a six-legged robot.

Table 8.3: Optimized rule base for first FLC (having eleven rules only) obtained using Approach 2 - ditch crossing module

distance \ relative angle	NL	NM	Z	PM	PL
VN		SL	LR		
NR	SM	SL			SM
FR	SM	MM			SM
VF	SM		SL	SM	

Table 8.4: Optimized rule base for second FLC (having eleven rules only) obtained using Approach 2 - ditch crossing module

distance \ relative angle	NL	NM	Z	PM	PL
VN		SL	LR		MM
NR		SL	LR		SM
FR	SM	MM			
VF	SM		SL		SM

Table 8.5: Optimized rule base for third FLC (having ten rules only) obtained using Approach 2 - ditch crossing module

distance \ relative angle	NL	NM	Z	PM	PL
VN	MM	SL			MM
NR		SL	LR		
FR		MM			SM
VF		SM		SM	SM

Table 8.6: Optimized rule base for fourth FLC (having eight rules only) obtained using Approach 2 - ditch crossing module

distance \ relative angle	NL	NM	Z	PM	PL
VN	MM			SL	MM
NR		SL			
FR	SM	MM	LR		
VF				SM	

Table 8.7: Optimized rule base for fifth FLC (having six rules only) obtained using Approach 2 - ditch crossing module

distance \ relative angle	NL	NM	Z	PM	PL
VN			LR		
NR		SL			
FR	SM	MM			
VF			SL	SM	

Table 8.8: Optimized rule base for sixth FLC (having eight rules only) obtained using Approach 2 - ditch crossing module

distance \ relative angle	NL	NM	Z	PM	PL
VN	MM				MM
NR	SM	SL			SM
FR					SM
VF		SM		SM	

Table 8.9: Optimized rule base for first FLC (having eleven rules only) obtained using Approach 2 - turning gait generation module

distance \ crab angle	NL	NM	Z	PM	PL
VN	VS		MM		VS
NR	SM		SL		SM
FR	MM	SL	LR		
VF			VL		SL

Table 8.10: Optimized rule base for second FLC (having eleven rules only) obtained using Approach 2 - turning gait generation module

distance \ crab angle	NL	NM	Z	PM	PL
VN	VS	SM	MM	SM	
NR		MM	SL	MM	
FR			LR		MM
VF	SL			LR	

Table 8.11: Optimized rule base for third FLC (having ten rules only) obtained using Approach 2 - turning gait generation module

distance \ crab angle	NL	NM	Z	PM	PL
VN	VS	SM	MM	SM	
NR		MM	SL		SM
FR			LR	SL	
VF				LR	

Table 8.12: Optimized rule base for fourth FLC (having four rules only) obtained using Approach 2 - turning gait generation module

distance \ crab angle	NL	NM	Z	PM	PL
VN				SM	VS
NR	SM				
FR					
VF	SL				

Table 8.13: Optimized rule base for fifth FLC (having seven rules only) obtained using Approach 2 - turning gait generation module

distance \ crab angle	NL	NM	Z	PM	PL
VN		SM	MM		VS
NR					
FR		SL	LR		
VF		LR		LR	

Table 8.14: Optimized rule base for sixth FLC (having twelve rules only) obtained using Approach 2 - turning gait generation module

distance \ crab angle	NL	NM	Z	PM	PL
VN		SM	MM	SM	
NR		MM	SL	MM	
FR		SL	LR	SL	
VF	SL	LR	VL		

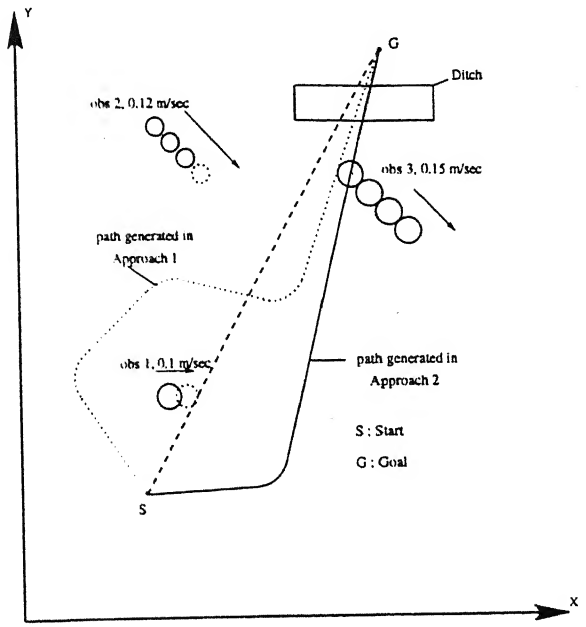


Figure 8.4: Generated paths in Approaches 1,2.

8.4 Summary

The problem of combined path and gait generation of a six-legged robot has been solved by using a genetic-fuzzy system, in this chapter. It is a complicated task and the traditional methods fail to find a complete solution. In the proposed algorithm, path planning and gait planning are done simultaneously by using the FLCs. There are seven FLCs running in parallel - one is for path generation and the remaining six are for gait generation. A GA is used to find optimized FLCs, off-line. A GA tries to find, through search, a set of good fuzzy rules from a manually constructed large rule base. As the variables are discrete in nature, a binary-coded GA is a natural choice for solving this type of problems. Once the optimized FLCs are obtained, those can be used on-line, to solve the real-world similar problems of combined path and gait planning, in an optimal sense. The GA-tuned FLCs are found to perform better than the manually constructed (author-defined) FLCs. The execution time of the proposed algorithm is found to be only 1.0 sec in a HP 9000/K200 machine. Thus, it is suitable for on-line implementation to solve this type of problems. Since GA is a population-based search and optimization technique, the chance of its solution for getting trapped into local minima is less.

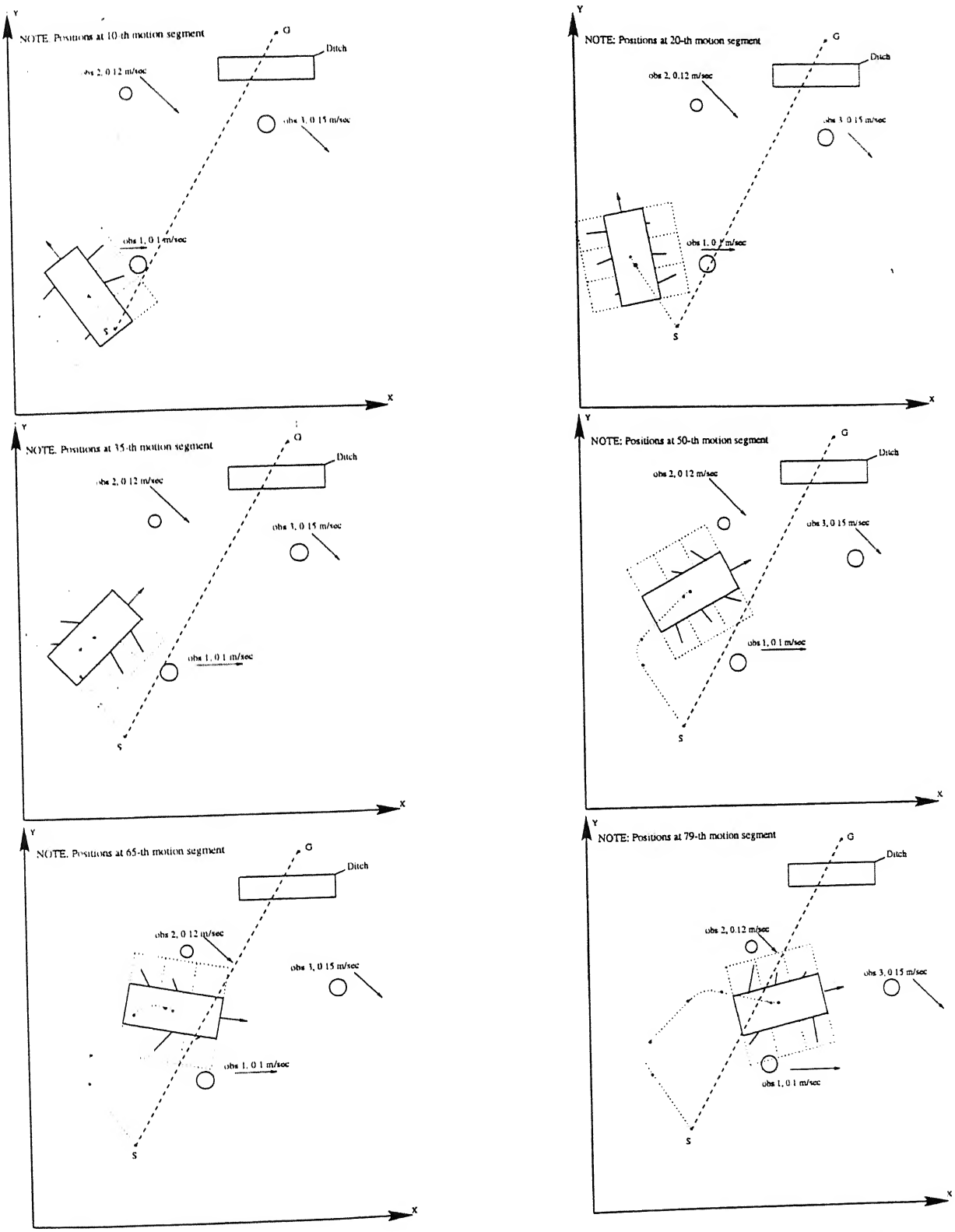


Figure 8.5: Generated path and gaits obtained using Approach 1 for test scenario 4 (Table 8.1)

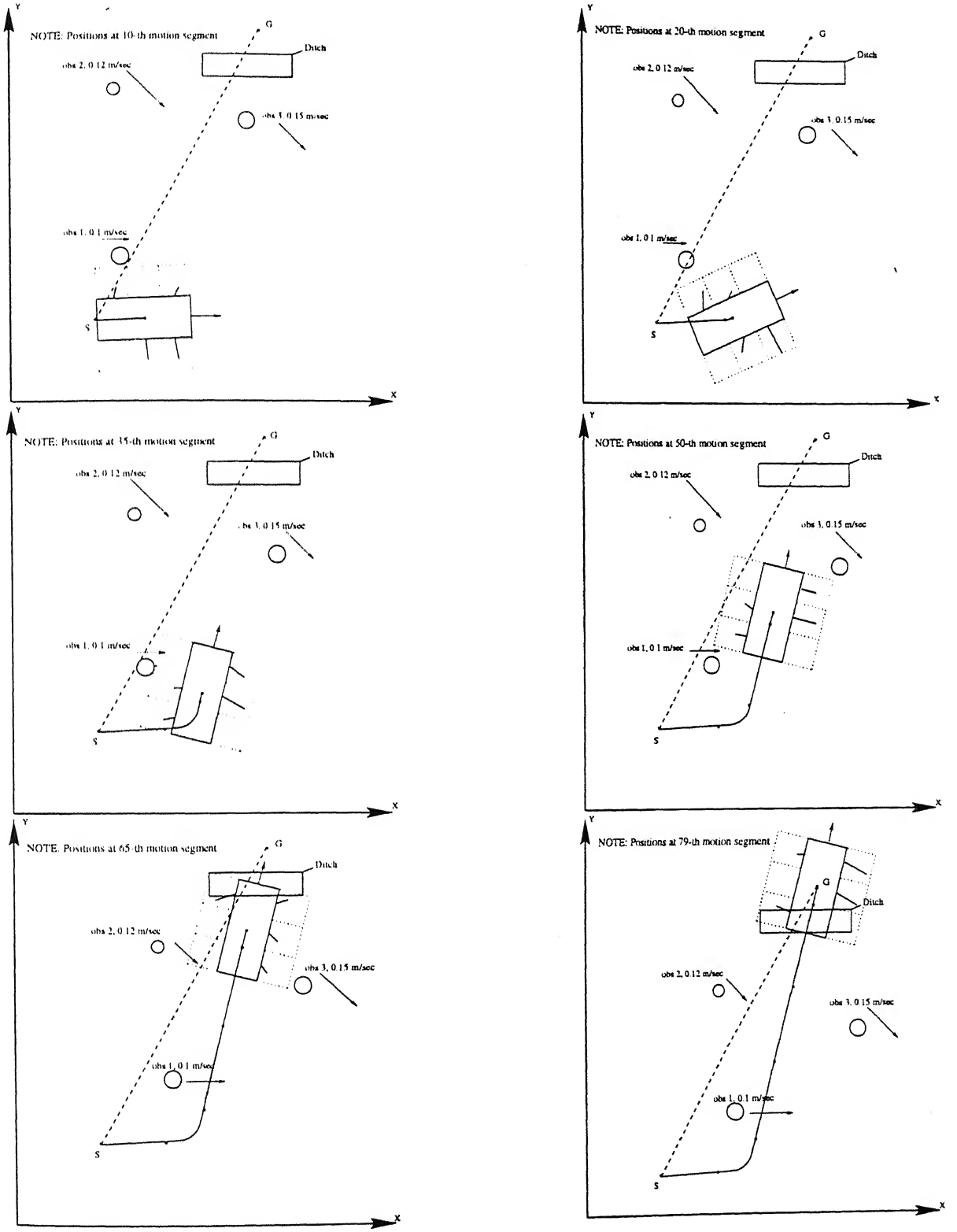


Figure 8.6: Generated path and gaits obtained using Approach 2 for test scenario 4 (Table 8.1)

Chapter 9

CONCLUDING REMARKS AND SCOPE FOR FUTURE WORK

9.1 Concluding Remarks

In the present work, path planning and gait planning problems of a legged robot have been solved by using the combined GA-Fuzzy approaches. Moreover, a genetic-fuzzy system has been developed to solve the problem of combined path and gait generation of a legged robot. From the above study, concluding remarks have been made as follows:

1. **Path Generation Among Static Obstacles:** The proposed *fuzzy-genetic algorithm*, as discussed in Chapter 3, is able to find time-optimal obstacle-free paths of a mobile robot in the presence of static obstacles. A fuzzy rule base approach has been used to find obstacle-free directions and a genetic algorithm is used as an optimizer to find the extent of travel along obstacle-free directions. The fuzzy logic approach has been introduced in the initial population creation and in the design of crossover and mutation operators in order to propagate useful building blocks through GA generations, a matter which is essential for an efficient use of GAs.

2. **Comparison of Fuzzy-Genetic Algorithm with Other Techniques:** The proposed *fuzzy-genetic algorithm* is found to perform better than a *steepest descent optimization method* along with a penalty function approach for solving the find-path problems of a mobile robot. This happens due to the fact that there is a chance of the solution for getting trapped into local minima in the steepest descent

method. On the other hand, since GA is a population-based search and optimization technique, the chance of its solution for getting trapped into local minima is less. Moreover, the solution of the proposed algorithm is similar to the path obtained using the best-known *tangent graph and A* algorithm.* It is important to note that the performance of the proposed algorithm can be further improved by proper tuning of rule base of the fuzzy logic controller. It is also interesting to note that the proposed algorithm requires linearly more computational time with an increase in number of intermediate control points, whereas the tangent graph and A* algorithm has quadratic computational complexity. Thus, the fuzzy-genetic algorithm is computationally faster than the combined tangent graph and A* algorithm.

3. **Path Generation Among Moving Obstacles:** The proposed *genetic-fuzzy approach*, as explained in Chapter 4, (in which a GA is used for proper tuning of knowledge base of the fuzzy logic controller) can successfully solve the navigation problem of a mobile robot in the presence of moving obstacles. Here, the navigation problems of a mobile robot have been solved by using a fuzzy logic controller. The navigation is based on sensor readings and future prediction of location of moving obstacles. As sensor readings are associated with imprecision and uncertainty, an FLC is a natural choice for solving this type of problems. In the proposed genetic-fuzzy approach, the tuning of knowledge base of an FLC is done off-line by using a GA as an optimization tool and once the optimal knowledge base of an FLC is obtained, a mobile robot can use it on-line, to navigate in real-world similar scenarios, in an optimal sense. Simulation results show that a GA-tuned FLC performs better than an author-defined FLC. It happens due to the fact that the author-defined knowledge base for an FLC may not be optimum always.

4. **Some Important Observations:** The performance of an FLC depends on its knowledge base which consists of the three components, namely *scaling factors, membership functions* and *rule set.* Rule base optimization involves the problem of dealing with discrete variables and GA is a powerful tool for solving this type of problems. As the task of finding optimum rule base as well as the shape of the membership function distribution are inter-dependent, it is more practical to consider the GA-tuned FLC where both rule base as well as scaling factors of the state variables have been optimized simultaneously (refer to Approach 5 of Section 4.3). It is also observed that optimizing rule base of an FLC is a rough-tuning process, whereas optimizing scaling factor of the state variables (which indicates

the base width of triangular membership functions) is a fine-tuning process. Thus, the optimized solutions are obtained during the optimization of rule base only and tuning of scaling factors cannot bring significant improvement in the accuracy of a solution.

5. **A Two-Stage Automatic Design of a Fuzzy Rule Base:** Although an FLC is an effective tool for dealing with imprecision and uncertainty, designing a proper knowledge base for an FLC is a difficult task. A two-stage design of a fuzzy rule base has been implemented in Section 4.3 (refer to Approach 6). In the proposed algorithm, no time is spent on manual construction of fuzzy rule base and it will be designed automatically by a GA through search. Due to iterative nature of a GA, the obtained rule base may contain some redundant rules which are removed from the rule base by a second stage GA-based tuning. This approach has been used to solve the navigation problem of a mobile robot in the presence of moving obstacles and the obtained results are almost similar to those obtained by the other methods, namely tuning rule base alone, tuning scaling factors and rule base in stages, tuning scaling factors and rule base simultaneously (refer to Section 4.3). However, it is a more flexible method in which a set of good rules for an FLC will be designed automatically by a GA and may be more efficient in solving more complex navigation problems.

6. **Gait Generation Problem Using a Genetic-Fuzzy Approach:** The proposed *genetic-fuzzy approach* is able to generate successfully near-optimal gait of a hexapod while crossing a ditch (refer to Chapter 6) or while taking a circular turn (refer to Chapter 7) separately. In all such applications, the GA-tuned FLCs (obtained by optimizing rule base alone) are found to perform better than the author-defined FLCs. Here, only the rule base of an FLC has been optimized because the performance of an FLC depends largely on its rule base and optimizing scaling factors of the state variables is a fine-tuning process.

7. **Combined Path and Gait Generation Using a Genetic-Fuzzy Approach:** The problem of combined path and gait generation of a hexapod (which is the most difficult problem compared to all problems discussed earlier) has also been solved by using the *genetic-fuzzy approach*. Here, there are six FLCs for controlling the six legs of a hexapod and one more FLC for generating the collision-free path of the robot. Besides generating a time-optimal path, the hexapod will have to generate its

gait in an optimal sense. Once the GA-tuned FLCs are obtained (tuning is done off-line), a hexapod can use those FLCs on-line, to generate near-optimal path and gait simultaneously. It is important to note that the traditional methods cannot solve the problem of combined path and gait generation effectively because it is a complicated optimization problem involving discrete variables, whereas the proposed algorithm is able to solve it. Moreover, the use of FLCs makes this approach computationally more tractable.

9.2 Scope for Future Work

- In the present work, path and gait generation of a six-legged robot has been studied in detail, by using the combined GA-Fuzzy approach. It can also be extended for the four-legged and eight-legged robots.

- In this study, an attempt is made to solve the problem of combined path and gait generation of a hexapod which is moving on a flat terrain by negotiating ditches and in the presence of moving obstacles. This approach can be extended further for the rough terrain also. In practice, the terrain may be uneven and possibly with some slope.

- In Section 4.3, the approach - *automatic two-stage design of fuzzy rules using GA* has been discussed in detail and its effectiveness has been tested on a number of path generation problems. The similar approach has not yet been implemented for solving the gait generation problem of a legged robot. In an extended version of the present work, a set of good fuzzy rules will be designed by a GA automatically to solve the problem of combined path and gait generation of a legged vehicle.

- In the present work, a six-legged robot will have to plan both its path as well as gait in such a way that it can reach its destination in minimum traveling time and by placing the minimum number of legs on the ground having the maximum average kinematic margin. Obviously, the problem can be better posed as a multi-objective optimization problem, which can be solved by using multi-objective GA to find multiple trade-off solutions.

- In this study, the issues related to the static stability of the vehicle have been considered. The present work can be extended further to consider the issues related to the dynamic stability of the vehicle.

Bibliography

[1] Arkin R.C. (1989), "Motor schema-based mobile robot navigation", *The Int. Jl. of Robotics Research*, 8, 92-112.

[2] Arkin R.C. (1990), "Integrating behavioral, perceptual, and world knowledge in reactive navigation", *Robotics and Autonomous Systems*, 6, 105-122.

[3] Arkin R.C. and MacKenzie D. (1994), "Temporal coordination of perceptual algorithms for mobile robot navigation", *IEEE Trans. on Robotics and Automation*, 10(3), 276-286.

[4] Ahuactzin J.M. et al. (1992), "Using genetic algorithms for robot motion planning", *Proc. of 10th European Conference on Artificial Intelligence - ECAI 92*, 671-675.

[5] Bagchi A. (1991), "Application of Fuzzy Logic for Collision Avoidance in Navigation and Manipulation in Dynamic Environments", PhD Thesis, M.E. Department, Indian Institute of Technology, Kanpur, India, 1991.

[6] Barraquand J., Langlois B. and Latombe J.C. (1992), "Numerical potential field techniques for robot path planning", *IEEE Trans. on Syst., Man and Cybern.*, 22, 224-241.

[7] Beaufrere B. and Zeghloul S. (1995), "A mobile robot navigation method using a fuzzy logic approach", *Robotica* 13, 437-448.

[8] Bessonov A. and Umnov N. (1973), "The analysis of gaits in six-legged vehicle according to their static stability", *Proc. Symp. Theory and Practice of Robots and Manipulators*, Udine, Italy, 1-9.

[9] Bonarini A. (1993), "Learning incomplete fuzzy rule sets for an autonomous robot", *Proc. of First European Congress on Fuzzy and Intelligent Technologies (EUFIT'93)*, Aachen, 69-75.

[10] Borenstein J. and Koren Y. (1989), "Real-time obstacle avoidance for fast mobile robots", *IEEE Trans. on Systems, Man and Cybernetics*, 19(5), 1179-1187.

[11] Borenstein J. and Koren Y. (1990), "Real-time obstacle avoidance for fast mobile robots in cluttered environments", *IEEE Intl. Conf. on Robotics and Automation*, Cincinnati, Ohio, 572-577.

[12] Borenstein J. and Koren Y. (1991), "The Vector Field Histogram - Fast Obstacle Avoidance for Mobile Robots", *IEEE Trans. on Robotics and Automation*, 7(3), 278-288.

[13] Brooks R.A. (1983), "Solving the find-path problem by good representation of free space", *IEEE Transactions on Systems, Man and Cybernetics*, SMC-13(3), 190-197.

[14] Brooks R.A. (1986), "A robust layered control system for a mobile robot", *IEEE Jl. on Robotics and Automation*, RA-2, 14-23.

[15] Brooks R.A. (1989), "A robot that walks; emergent behaviors from a carefully evolved network", *Neural Computation*, 1(2), 355-363.

[16] Canny J. and Reif J. (1987), "New lower bound techniques for robot motion planning problems", *Proc. of IEEE Symp. on Foundations of Computer Science*, 49-60.

[17] Chan R.H.T., Tam P.K.S., Leung D.N.K. (1994), "Solving the motion planning problem by using neural networks", *Robotica*, 12, 323-333.

[18] Chen C. and Kumar V. (1996), "Motion Planning of Walking Robots in Environments with Uncertainty", *Proc. of IEEE Intl. Conf. on Robotics and Automation*, Minneapolis, Minnesota, 3277-3282.

[19] Chien S.A., Gervasio M.T. and DeJong G.F. (1991), "On becoming decreasing reactive: Learning to deliberate minimally", *Proc. 8th Intl. Workshop Machine learning*, Chicago, 288-292.

[20] Chiel H.J., Beer R.D., Quinn D. and Espenschied K.S. (1992), "Robustness of a distributed neural network controller for locomotion in a hexapod robot", *IEEE Trans. on Robotics and Automation*, 8(3), 293-303.

[21] Cooper M.G. and Vidal J.J. (1993), "Genetic design of fuzzy logic controllers", *Proc. of Second International Conference on Fuzzy Theory and Technology (FTT'93)*, Durham.

[22] Cordon O., Herrera F., Lozano M. (1997), "On the combination of fuzzy logic and evolutionary computation: a short review and bibliography", In: *Fuzzy Evolutionary Computation*, W. Pedrycz (Ed.), Kluwer Academic Press, Norwell, MA, 33-44.

[23] Carse B. and Fogarty T.C. (1994), "A Fuzzy classifier system using the Pittsburgh approach", *Proc. of Third Conference on Parallel Problem Solving from Nature, Lecture Notes on Computer Science vol. 866*, Springer-Verlag, 260-269.

[24] Donnart J. and Meyer J. (1996), "Learning Reactive and Planning Rules in a Motivationally Autonomous Animat", *IEEE Trans. on Systems, Man, and Cybernetics - Part B*, 26(3), 381-395.

[25] Dorigo M. and Schnepf U. (1993), "Genetics-Based Machine Learning and Behavior-based Robotics: A new synthesis", *IEEE Trans. on Systems, Man and Cybernetics*, 23(1), 141-153.

[26] Deb K. (1995), "Optimization for Engineering Design", Prentice-Hall of India Pvt. Ltd., New Delhi, India.

[27] Deb K. (1998), "Genetic Algorithm in Search and Optimization: The Techniques and Applications", *Proc. of International Workshop on Soft Computing and Intelligent Systems*, ISI, Calcutta, India, 58-87.

[28] Deb K. and Agrawal R.B. (1995), "Simulated Binary Crossover for Continuous Search Space", *Complex Systems*, 9, 115-148.

[29] Deb K. and Agrawal S. (1999), "Understanding interactions among genetic algorithm parameters", *Proceedings of Foundations of Genetic Algorithms*, W. Banzhaf and C. Reeves (Eds.), Morgan Kaufmann, 265-286.

[30] Deb K., Pratihar D.K. and Ghosh A. (1998), "Learning to avoid moving obstacles optimally for mobile robots using a GA-Fuzzy approach", *Proc. of the Fifth International Conference on Parallel Problems Solving from Nature*, Amsterdam, The Netherlands, 583-592.

[31] Eiji Nawa N., Hashiyama T., Furuhashi T., Uchikawa Y. (1997), "Fuzzy Logic Controllers Generated by Pseudo-Bacterial Genetic Algorithm with Adaptive Operator", *Proc. of IEEE International Conference on Neural Networks - ICNN'97*, Houston. USA, 2408-2413.

[32] Enbutsu I. et al. (1991), "Fuzzy Rule Extraction from a Multilayered Neural Network", *Proc. of International Joint Conference on Neural Networks*, Seattle, WA. 461-465.

[33] Erdmann M. and Lozano-Pérez T. (1986), "On multiple moving objects", *Proc. of IEEE Intl. Conf. on Robotics and Automation*, 1419- 1424.

[34] Eshelman L.J. and Schaffer J.D. (1993), "Real-coded genetic algorithms and interval schemata", *Foundations of Genetic Algorithms*, D. Whitley (Ed.), 187-202.

[35] Ferrari C. and Chemello G. (1990), "Coupling Fuzzy Logic Techniques with Evidential Reasoning for Sensor data Interpretation", *Proc. of Intelligent Autonomous Systems*, 965-971.

[36] Fiorini P. and Shiller Z. (1993), "Motion planning in dynamic environments using the relative velocity paradigm", *Proc. of IEEE Conf. on Robotics and Automation*, 560-565.

[37] Fraichard T. and Laugier C. (1993), "Path-velocity decomposition revisited and applied to dynamic trajectory planning", *Proc. of IEEE Int. Conf. on Robotics and Automation*, 40-45.

[38] Fujimura K. (1995), "Time-minimum routes in time-dependent networks", *IEEE Trans. on Robotics and Automation*, 11(3), 343-351.

[39] Fujimura K. and Samet H. (1988), "Accessibility : a new approach to path planning among moving obstacles", *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition*, Ann Arbor, MI, 803-807.

[40] Fujimura K. and Samet H. (1989), "A hierarchical strategy for path planning among moving obstacles", *IEEE Trans. on Robotics and Automation*, 5(1), 61-69.

[41] Gat E. (1991), "Robust, task-directed, reactive control of autonomous mobile robots", PhD Thesis,Virginia Polytechnic Institute and State University.

[42] Garis H. (1990), "Genetic Programming: Building Nanobrains with Genetically Programmed Neural Network Modules", *Proc. of the Intl. Joint Conference on Neural Networks*, III-511-III-516.

[43] Goldberg D.E. (1989), "Genetic Algorithms in Search, Optimization, and Machine Learning", Addison-Wesley, Reading, Mass., USA.

[44] Goldberg D.E. and Deb K. (1991), "A comparison of selection schemes used in genetic algorithms", *Foundations of Genetic Algorithms*, G.J.E. Rawlins (Ed.), 69-93.

[45] Goldberg D.E., Deb K., Clark J.H. (1992), "Genetic algorithms, noise, and the sizing of populations", *Complex Systems*, 6, 333-362.

[46] Griswold N.C. and Eem J. (1990), "Control for mobile robots in the presence of moving objects", *IEEE Trans. on Robotics and Automation*, 6(2), 263-268.

[47] Gruau F. and Quatramaran K. (1996), "Cellular Encoding for Interactive Evolutionary Robotics", *Proc. of Micro-Robot World Cup Soccer Tournament*, 158-180.

[48] Harvey I., Husbands P., Cliff D. (1993), "Genetic Convergence in a Species of Evolved Robot Control Architectures", *Proc. of Fifth International Conference on Genetic Algorithms*, 636.

[49] Harp S.A. and Samad T. (1991), "Genetic Synthesis of Neural Network Architecture", *Handbook of GAs*, L. Davis (Ed.), Van Nostrand Reinhold, N.Y., 202-221.

[50] Holland J. (1975), "Adaptation in Natural and Artificial Systems", The University of Michigan Press, Ann Arbor, USA.

[51] Herrera F., Herrera-Viedma E., Lozano M., Verdegay J.L. (1994), "Fuzzy tools to improve genetic algorithms", *Proc. of the Second European Congress on Intelligent Techniques and Soft Computing*, Aachen, 1532-1539.

[52] Herrera F., Lozano M., Verdegay J.L. (1995a), "Tuning fuzzy logic controllers by genetic algorithms", *International Journal of Approximate Reasoning*, 12, 293-315.

[53] Herrera F., Lozano M., Verdegay J.L. (1995b), "A learning process for fuzzy control rules using genetic algorithms", *Technical Report DECSAI-95108*, University of Granada, Department of Computer Science and Artificial Intelligence, Granada, Spain.

[54] Hildebrand M. (1967), "Symmetrical Gaits of Horses", *Science*, 150, 701-708.

[55] Hirose S. (1984), "A study of design and control of a quadruped walking vehicle", *International Journal of Robotics Research*, 3(2), 113-133.

[56] Ishigami H., Fukuda T., Shibata T., Arai F. (1995), "Structure optimization of fuzzy neural networks by genetic algorithms", *Fuzzy Sets and Systems*, 71(3), 257-264.

[57] Jiménez M.A. and Santos P.G. (1997), "Terrain-Adaptive Gait for Walking Machines", *The International Journal of Robotics Research*, 16(3), 320-339.

[58] Kant K. and Zucker S.W. (1984), "Trajectory planning problems I: Determining velocity along a fixed path", *Proc. of 7-th IEEE Intl. Conf. on Pattern Recognition*, Montreal, 196-198.

[59] Kant K. and Zucker S.W. (1986), "Towards efficient planning: The path velocity decomposition", *The Int. Jl. of Robotics Research*, 5, 72-89.

[60] Kant K. and Zucker S.W. (1988), "Planning collision-free trajectories in time-varying environments: A two-level hierarchy", *Proc. of IEEE Intl. Conf. on Robotics and Automation*, 1644-1649.

[61] Kolodner J.L. (1990), "An introduction to case-based reasoning", *Tech. Rep. GIT-ICS-90/19*, School Inform. Comput. Sci., Georgia Inst. Technol., Atlanta, GA.

[62] Kosko B. (1994), "Neural Networks and Fuzzy Systems", Prentice-Hall, New Delhi, India.

[63] Kargupta H., Deb K., Goldberg D.E. (1992), "Ordering genetic algorithms and deception", *Proc. of Parallel Problem Solving from Nature II*, 47-56.

[64] Klir G.J. and Yuan B. (1997), "Fuzzy Sets and Fuzzy Logic: Theory and Applications", Prentice-Hall of India Pvt. Ltd., New Delhi, India.

[65] Karr C. (1991a), "Genetic algorithms for fuzzy controllers", *AI Expert*, 26-33.

[66] Karr C. (1991b), "Applying genetics", *AI Expert*, 38-43.

[67] Khatib O. (1986), "Real-Time Obstacle Avoidance for Manipulators and Mobile Robots", *The International Journal of Robotics Research*, 5(1), 90-98.

[68] Khosla P. and Volpe R. (1988), "Superquadric artificial potentials for obstacles avoidance and approach", *Proc. of IEEE Conf. on Robotics and Automation*, 1778-1784.

[69] Kim B. and Shin K. G. (1984), "An efficient minimum-time robot path planning under realistic constraints", *Proceedings of American Control Conference*, 296-303.

[70] Kim J.O. and Khosla P.K. (1992), "Real-time obstacle avoidance using harmonic potential functions", *IEEE Trans. on Robotics and Automation*, 8(3), 338-349.

[71] Koren Y. and Borenstein J. (1991), "Potential field methods and their inherent limitations for mobile robot navigations", *Proc. of IEEE Intl. Conference on Robotics and Automation*, 1398-1404.

[72] Kugushev E.I. and Jaroshevskij V.S. (1975), "Problems of selecting a gait for an integrated locomotion robot", *Proc. of 4th Intl. Conf. Artificial Intelligence*, Tbilisi, Georgian SSR, USSR, 789-793.

[73] Kumar V.R. and Waldron K.J. (1989), "Adaptive gait control for a walking robot", *Journal of Robotic Systems*, 49-76.

[74] Kwak S.H. (1984), "A simulation study of free gait algorithms for omnidirectional control of hexapod walking machines", M.S. thesis, The Ohio State University, Columbus, OH.

[75] Kyriakopoulos K.J. and Saridis G.N. (1991), "Collision avoidance of mobile robots in non-stationary environments", *Proc. IEEE Intl. Conf. on Robotics and Automation*, Sacramento,California, 904-909.

[76] Lamadrid J.G. and Gini M.L. (1990), "Path tracking through uncharted moving obstacles", *IEEE Trans. on Systems, Man and Cybernetics*, 20(6), 1408-1422.

[77] Lamadrid J.G. (1994), "Avoidance of obstacles with unknown trajectories: Locally optimal paths and periodic sensor readings", *The Int. Jl. of Robotics Research*, 496-507.

[78] Latombe J.C. (1991), "Robot Motion Planning", Kluwer Academic Publishing, Norwell, MA.

[79] Leung C.H. et al. (1994), "A genetic solution for the motion of wheeled robotic systems in dynamic environments", *Proc. of Intl. Conf. on Control'94*, Coventry.

[68] Khosla P. and Volpe R. (1988), "Superquadric artificial potentials for obstacles avoidance and approach", *Proc. of IEEE Conf. on Robotics and Automation*, 1778-1784.

[69] Kim B. and Shin K. G. (1984), "An efficient minimum-time robot path planning under realistic constraints", *Proceedings of American Control Conference*, 296-303.

[70] Kim J.O. and Khosla P.K. (1992), "Real-time obstacle avoidance using harmonic potential functions", *IEEE Trans. on Robotics and Automation*, 8(3), 338-349.

[71] Koren Y. and Borenstein J. (1991), "Potential field methods and their inherent limitations for mobile robot navigations", *Proc. of IEEE Intl. Conference on Robotics and Automation*, 1398-1404.

[72] Kugushev E.I. and Jaroshevskij V.S. (1975), "Problems of selecting a gait for an integrated locomotion robot", *Proc. of 4th Intl. Conf. Artificial Intelligence*, Tbilisi, Georgian SSR, USSR, 789-793.

[73] Kumar V.R. and Waldron K.J. (1989), "Adaptive gait control for a walking robot", *Journal of Robotic Systems*, 49-76.

[74] Kwak S.H. (1984), "A simulation study of free gait algorithms for omnidirectional control of hexapod walking machines", M.S. thesis, The Ohio State University, Columbus, OH.

[75] Kyriakopoulos K.J. and Saridis G.N. (1991), "Collision avoidance of mobile robots in non-stationary environments", *Proc. IEEE Intl. Conf. on Robotics and Automation*. Sacramento,California, 904-909.

[76] Lamadrid J.G. and Gini M.L. (1990), "Path tracking through uncharted moving obstacles", *IEEE Trans. on Systems, Man and Cybernetics*, 20(6), 1408-1422.

[77] Lamadrid J.G. (1994), "Avoidance of obstacles with unknown trajectories: Locally optimal paths and periodic sensor readings", *The Int. Jl. of Robotics Research*, 496-507.

[78] Latombe J.C. (1991), "Robot Motion Planning", Kluwer Academic Publishing, Norwell, MA.

[79] Leung C.H. et al. (1994), "A genetic solution for the motion of wheeled robotic systems in dynamic environments", *Proc. of Intl. Conf. on Control'94*, Coventry, UK, 760-764.

[80] Lee J. and Bein Z. (1990), "Collision-free trajectory control for multiple robots based on neural optimization network", *Robotica*, 8(3), 185-194.

[81] Lee J. and Song S. (1991), "Path Planning and Gait of Walking Machine in an Obstacle-Strewn Environment", *Journal of Robotic Systems*, 8(6), 801-827.

[82] Lee M.A. and Takagi H. (1993a), "Dynamic control of genetic algorithms using fuzzy logic techniques". *Proc. of the Fifth International Conference on Genetic Algorithms*, Morgan Kaufmmann, San Mateo, 76-83.

[83] Lee M.A. and Takagi H. (1993b), "Embedding a priori knowledge into an integrated fuzzy system design method based on genetic algorithms", *Proc. of Fifth International Fuzzy Systems Association World Congress (IFSA'93)*, Seoul, 1293-1296.

[84] Lewis M.A., Fagg A.H. and Solidum A. (1992), "Genetic Programming Approach to the Construction of a Neural Network for Control of a Walking Robot". *Proc. of IEEE Intl. Conf. on Robotics and Automation*, Nice, France, 2618-2623.

[85] Li W. (1997), "A Method for Design of a Hybrid Neuro-Fuzzy Control Syatem Based on Behavior Modeling", *IEEE Trans. on Fuzzy Systems*, 5(1), 128-137.

[86] Lin L.J. (1993), "Scaling up reinforcement learning for robot control", *Proc. 10th Intl. Conf. Machine Learning*, San Mateo, CA: Morgan Kaufmann, 182-189.

[87] Lin H.S., Xiao J., Michalewicz Z. (1994), "Evolutionary navigator for a mobile robot", *Proc. of IEEE Intl. Conf. on Robotics and Automation*, 2199-2204.

[88] Leven D. and Sharir M. (1987), "Planning a purely translational motion for a convex object in two-dimensional space using generalized voronoi diagrams". *Discrete Comput. Geometry*, 2, 9-31.

[89] Liska J. and Melsheimer S.S. (1994), "Complete design of fuzzy logic systems using genetic algorithms", *Proc. of the Third IEEE International Conference on Fuzzy Systems*, IEEE, Piscataway NJ, 1377-1382.

[90] Liu Y.H. and Arimoto S. (1991), "Proposal of tangent graph and extended tangent graph for path planning of mobile robots", *Proc. of IEEE International Conference on Robotics and Automation*, 312-317.

[91] Liu Y. H. and Arimoto S. (1992), "Path planning using a tangent graph for mobile robots among polynomial and curved obstacles", *International Journal of Robotics Research*, 11(4), 376-382.

[92] Liu Y. H. and Arimoto S. (1995), "Finding the shortest path of a disc among polygonal obstacles using a radius-independent graph", *IEEE Transactions on Robotics and Automation*, 11(5), 682-691.

[93] Lozano-Pérez T. and Wesley M.A. (1969), "An algorithm for planning collision-free paths among polyhedral obstacles", *ACM*, 22(10), 560-570.

[94] Lozano-Pérez T. (1983), "Spatial planning: A configuration space approach", *IEEE Transactions on Computers*, C-32(2), 108-120.

[95] Lozano-Pérez T. (1987), "A simple motion planning algorithm for general robot manipulators", *IEEE Journal of Robotics and Automation*, 3(3), 224-238.

[96] Magdalena L. and Velasco J.R. (1996), "Fuzzy rule-based controllers that learn by evolving their knowledge base", In: *Genetic Algorithms and Soft Computing*, F. Herrera and J.L. Verdegay (Eds.), Physica-Verlag, 172-201.

[97] Mahadevan S. (1992), "Enhancing transfer in reinforcement learning by building stochastic models of robot actions", *Proc. 9th Intl. Workshop Machine learning*, San Mateo, CA: Morgan Kaufmann, 290-299.

[98] Mamdani E.H. and Assilian S. (1975), "An experiment in linguistic synthesis with a fuzzy logic controller", *International Journal of Man-Machine Studies*, 7(1), 1-13.

[99] Martinez A., Tunstel E., Jamshidi M. (1994), "Fuzzy logic based collision avoidance for a mobile robot", *Robotica*, 12, 521-527.

[100] Merriam-Webster (1985), "Webster's Ninth New Collegiate Dictionary", Merriam-Webster, Springfield, Massachusetts.

[101] McGhee R.B. and Frank A.A. (1968), "On the stability properties of quadruped creepings gaits", *Mathematical Biosciences*, 3, 331-351.

[102] McGhee R.B. and Iswandhi G.I. (1979), "Adaptive locomotion of a multilegged robot over rough terrain", *IEEE Trans. on Systems, Man, and Cybernetics*, SMC-9(4), 176-182.

[103] Millán J.R. (1996), "Rapid, Safe, and Incremental Learning of Navigation Strategies", *Proc. of IEEE Trans. on Systems, Man, and Cybernetics - Part B*, 26(3), 408-420.

[104] Michalewicz Z. (1994), "Genetic Algorithms + Data Structures = Evolution Programs", Springer-Verlag, New Yord, Second Edition.

[105] Min B. and Bien Z. (1992), "Neural computation for adaptive gait control of the quadruped over rough terrain", *Proc. of IEEE Intl. Conf. on Robotics and Automation*, Nice, France, 2612-2617.

[106] Mitchell J.S.B. (1988), "An algorithmic approach to some problems in terrain navigation", *Artificial Intelligence*, 37, 171-201.

[107] Mitchell T.M. (1990), "Becoming increasingly reactive", *Proc. Nat. Conf. Artificial Intelligence*, Boston, MA, 1051-1058.

[108] Muybridge E. (1899), "Animals in Motion", New Dover Edition, Dover Publications Inc., New York.

[109] Muybridge E. (1901), "The Human Figure in Motion", Dover Publications Inc., New York.

[110] Nam Y.S., Lee B.H. and Ko N.Y. (1995), "An Analytic Approach to Moving Obstacle Avoidance Using an Artificial Potential Field", *Proc. IEEE/RSJ Int. Conf. IROS*, 482-487.

[111] Nam Y.S., Lee B.H. and Kim M. S. (1996), "View-Time Based Moving Obstacle Avoidance Using Stochastic Prediction of Obstacle Motion", *Proc. IEEE Int. Conf. on Robotics and Automation*, Minneapolis, Minnesota, 1081-1086.

[112] Newman W.S. and Hogan N. (1987), "High speed robot control and obstacle avoidance using dynamic potential functions", *Proc. IEEE Int. Conf. on Robotics and Automation*, 14-24.

[113] Ng K.C. and Lee Y. (1994), "Design of sophisticated fuzzy logic controllers using genetic algorithms", *Proc. of Third IEEE International Conference on Fuzzy Systems (FUZZ-IEEE'94)*, Orlando, 1708-1712.

[114] Nilsson N.J. (1969), "A Mobile Automation: An Application of Artificial Intelligence Techniques", *Proc. of the 1st International Joint Conference on Artificial Intelligence (IJCAI)*, Washington D.C., 509-520.

[115] Okutomi M. and Mori M. (1986), "Decision of robot movement by means of a potential field", *Advanced Robotics*, 1(2), 131-141.

[116] O'Dunlaing C., Sharir M. and Yap C. (1986), "Generalized voronoi diagrams for a ladder I: Topological analysis", *Comm. Pure Appl. Math.*, 39, 423-483.

[117] O'Dunlaing C., Sharir M. and Yap C. (1987), "Generalized voronoi diagrams for a ladder II: Efficient construction of the diagram", *Algorithmica*, 2, 27-59.

[118] Ozguner F., Tsai S.J. and McGhee R.B. (1984), "An approach to the use of terrain-preview information by a hexapod walking machine", *International Journal of Robotics Research*, 3(2), 134-146.

[119] Pal P.K. and Jayarajan K. (1990), "A free gait for generalized motion", *IEEE Trans. on Robotics and Automation*, RA-6(5), 597-600.

[120] Pal P.K. and Jayarajan K. (1991), "Generation of free gait - a graph search approach", *IEEE Trans. on Robotics and Automation*, RA-7(3), 299-305.

[121] Pal P.K., Mahadev V. and Jayarajan K. (1994), "Gait Generation for a Six-legged Walking Machine through Graph Search", *Proc. of IEEE International Conference on Robotics and Automation*, San Diego, California, 1332-1337.

[122] Patterson M.R., Reidy J.J. and Brownstein B.J. (1983), "Guidance and actuation techniques for an adaptively controlled vehicle", *Final Tech. Rept.*, Columbus , Ohio: Battelle Columbus Laboratories.

[123] Pratihar D.K., Deb K. and Ghosh A. (1998a), "Planning Crab Gaits of a Six-legged Robot Using GA-Fuzzy Approach", *Proc. of the International Conference on Information Technology*, Bhubaneswar, India, 221-226.

[124] Pratihar D.K., Deb K. and Ghosh A. (1998b), "Mobile Robot Navigation Using GA-Fuzzy Approach", *Proc. of ICTACEM'98*, IIT Kharagpur, India.

[125] Pratihar D.K., Deb K. and Ghosh A. (1999a), "A Genetic-Fuzzy Approach for Mobile Robot Navigation Among Moving Obstacles", *International Journal of Approximate Reasoning*, 20, 145-172.

[126] Pratihar D.K., Deb K. and Ghosh A. (1999b), "Design of a Genetic-Fuzzy System for Planning Crab Gaits of a Six-legged Robot", *Journal of Computing and Information Technology*, special issue on Evolutionary Computation, (in press).

[127] Pratihar D.K., Deb K. and Ghosh A. (1999c), "Fuzzy-Genetic Algorithms and Time-optimal Obstacle-free Path Generation for Mobile Robots", *Engineering Optimization*, (in press).

[128] Pratihar D.K., Deb K. and Ghosh A. (1999d), "Optimal Turning Gait of a Six-legged Robot Using a GA-Fuzzy Approach", *Artificial Intelligence for Engineering Design, Analysis and Manufacturing - AIEDAM*, (under review).

[129] Pratihar D.K., Deb K. and Ghosh A. (1999e), "Optimal Path and Gait Generations Simultaneously of a Six-legged Robot Using a GA-Fuzzy Approach", *IEEE Trans. on Systems, Man and Cybernetics*, (under review).

[130] Pratihar D.K., Deb K. and Ghosh A. (1999f), "Design of a Genetic-Fuzzy System for Planning Optimal Path and Gait Simultaneously of a Six-legged Robot", Accepted for presentation and publication in the *Proc. of Genetic and Evolutionary Computation - GECCO 99*, to be held on 13-17 July, 1999, at Orlando, USA.

[131] Pin F.G., Watanabe Y. (1994), "Navigation of mobile robots using a fuzzy behaviorist approach and custom-designed fuzzy inferencing boards", *Robotica*, 12, 491-503.

[132] Pinchard O. et al. (1995), "A genetic algorithm for outdoor robot path planning", In: *Intelligent Autonomous Systems (IAS-4)*, U. Rembold (Ed.), Karlsruhe, Germany, 413-419.

[133] Radcliffe N. J. (1991), "Forma Analysis and Random Respectful Recombination". Proc. of the Fourth Intl. Conference on Genetic Algorithms, 222-229.

[134] Ram A., Arkin R.C., Moorman K. and Clark R.J. (1997), "Case-based Reactive Navigation: A Method for On-line Selection and Adaptation of Reactive Robotic Control Parameters", *IEEE Trans. on Systems, Man and Cybernetics- part B: Cybernetics*, 27(3), 376-394.

[135] Rahman Z.U. (1977), "Optimization of creeping gaits for quadruped leveled vehicles", Ph.D. dissertation, Auburn University, Alabama.

[136] Reif J. H. (1979), "Complexity of the Movers' Problem and Generalizations", *Proceedings of the 20th IEEE Symposium on the Foundations of Computer Science* (FOCS), 421-427.

[137] Reif J. and Sharir M. (1985), "Motion planning in the presence of moving obstacles", *Proc. of IEEE Symp. on Foundations of Computer Science*, 144-154.

[138] Reister D. B. and Lenhart S. M. (1995), "Time-Optimal Paths for High-Speed Maneuvering", *The International Journal of Robotics Research*, 14(2), 184-194.

[139] Rich E. (1983), "Artificial Intelligence", McGraw-Hill, Japan.

[140] Rimon E. and Koditschek D.E. (1992), "Exact robot navigation using artificial potential functions", *IEEE Trans. on Robotics and Automation*, 8(5), 501-518.

[141] Schwartz J. T. and Sharir M. (1983a), "On the Piano Movers' Problem: I. The Case if a Two-Dimensional Rigid Polygonal Body Moving Amidst Polygonal Barriers", *Communications on Pure and Applied Mathematics*, 36, 345-398.

[142] Schwartz J. T. and Sharir M. (1983b), "On the Piano Movers' Problem: II. General Techniques for Computing Topological Properties of Real Algebraic Manifolds", *Advances in Applied Mathematics*, 4, 298–351.

[143] Schwartz J. T. and Sharir, M. (1983c), "On the Piano Movers' Problem: III, Coordinating the Motion of Several Independent Bodies: The Special Case of Circular Bodies Moving Amidst Polygonal Barriers", *The International Journal of Robotics Research*, 2(3), 46-75.

[144] Schwartz J. T. and Sharir M. (1983d), "On the Piano Movers' Problem: V. The Case of a Rod Moving in Three-Dimensional Space Amidst Polyhedral Obstacles", *Communications on Pure and Applied Mathematics*, 37, 815-848.

[145] Sharir M. and Ariel-Sheffi E. (1983), "On the Piano Movers' Problem: IV. Various Decomposable Two-Dimensional Planning Problems", *Communications on Pure and Applied Mathematics*, 37, 479-493.

[146] Sharma R. (1992), "A probabilistic framework for dynamic motion planning in partially known environments", *Proc. of IEEE Int. Conf. on Robotics and Automation*, Nice, France, 2459-2464.

[147] Shih C.L. and Klein C.A. (1993), "An adaptive gait for legged walking machines over rough terrain", *IEEE Trans. on Systems, Man and Cybernetics*, 23(4), 1150-1155.

[148] Sindall J.N. (1964), "The wave mode of walking locomotion", *Jl. Terramechanics* 1, 54-73.

[149] Slack M.G. and Miller D.P. (1987), "Path planning through time and space in dynamic domains", *Proc. of the Intl. Joint Conf. on Artificial Intelligence*, 1067-1070.

[150] Slotine J. J. and Yang H. S. (1989), "Improving the efficiency of time-optimal path-following algorithms", *IEEE J. Robotics and Automation*, 5(1), 118-124.

[151] Song S. and Waldron K.J. (1989), "Machines That Walk - The Adaptive Suspension Vehicle", The MIT Press, Cambridge, USA.

[152] Song S. and Waldron K.J. (1987), "An analytical approach for gait study and its applications on wave gaits", *The International Journal of Robotics Research*, 6(2), 60-71.

[153] Song S.M. (1984), "Kinematic optimal design of a six-legged walking machine", Ph.D. thesis, The Ohio State University, Columbus, Ohio.

[154] Spears W.M. and De Jong K.A. (1991), "An anlysis of multi-point crossover", *Foundations of Genetic Algorithms*, G.J.E. Rawlins (Ed.), 301-315.

[155] Satyadas A. and Krishnakumar K. (1994), "GA-optimized fuzzy controller for spacecraft attitude control", *Proc. of Third IEEE International Conference on Fuzzy Systems (FUZZ-IEEE'94)*, Orlando, 1979-1984.

[156] Srinivas N. and Deb K. (1995), "Multiobjective function optimization using non-dominated sorting genetic algorithms", *Evolutionary Computation*, 2(3), 221-248.

[157] Shin C.L., Lee T.T., Gruver W.A. (1990), "A unified approach for robot motion planning with moving polyhedral obstacles", *IEEE Trans. on Systems, Man and Cybernetics*, 20(4), 903-915.

[158] Suh S. and Shin K. (1988), "A variational dynamic programming approach to robot-path planning with a distance-safety criterion", *IEEE Jl. Robotics and Automation*, 4(3), 334-349.

[159] Sun S.S. (1974), "A theoretical study of gaits for a legged locomotion system", Ph.D. thesis, The Ohio State University, Columbus, Ohio.

[160] Tani J. (1996), "Model-Based Learning for Mobile Robot Navigation from the Dynamical Systems Perspective", *IEEE Trans. on Systems, Man, and Cybernetics - Part B*, 26(3), 421-436.

[161] Takagi H. and Hayashi I. (1991), "NN-Driven Fuzzy Reasoning", *International Journal of Approximate Reasoning*, 5(3), 191-212.

[162] Takeuchi T., Nagai Y. and Enomoto Y. (1988), "Fuzzy Control of a Mobile Robot for Obstacle Avoidance", *Information Sciences*, 45(2), 231-248.

[163] Thrift P. (1991), "Fuzzy logic synthesis with genetic algorithms", *Proc. of Fourth International Conference on Genetic Algorithms (ICGA'91)*, 509-513.

[164] Tsai S.J. (1983), "An experimental study of a binocular vision system for rough terrain locomotion of a hexapod walking robot", Ph.D. thesis, The Ohio State University, Columbus, Ohio.

[165] Tomovic R. and Karplus W.J. (1961), "Land locomotion simulation and control", *Proc. of Third Intl. Analogue Computation Meeting*, Opatija, Yugoslavia, 385-390.

[166] Vestli S.J. et al. (1993), "Integration of path planning, sensing and control in mobile robotics", *Proc. of IEEE Intl. Conference on Robotics and Automation*, 3, 243-248.

[167] Warren C. H. (1989), "Global Path Planning Using Artificial Potential Fields", *Proceedings of IEEE International Conference on Robotics and Automation*, 316-321.

[168] Wang S.L. (1983), "The study of a hexapod walking vehicle's maneuverability over level ground and obstacles and its computer simulation", M.S. thesis, The Ohio State University, Columbus, Ohio.

[169] Wettergreen D., Pangels H., and Bares J. (1995), "Behavior-based gait execution for the Dante II walking robot", *Proc. of IROS*, Pittsburgh, PA, 274-279.

[170] Wright A. (1991), "Genetic algorithms for real parameter optimization", *Foundations of Genetic Algorithms*, G.J.E. Rawlins (Ed.), 205-220.

[171] Xiao J., Michalewicz Z., Zhang L., Trojanowski K. (1997), "Adaptive evolutionary planner/navigator for mobile robots", *IEEE Trans. on Evolutionary Computation.* 1(1), 18-28.

[172] Xu H.Y. and Vukovich G. (1993), "A fuzzy genetic algorithm with effective search and optimization", *Proc. of 1993 International Joint Conference on Neural Networks.* 2967-2970.

[173] Xu H.Y., Vukovich G., Ichikawa Y., Ishii Y. (1994), "Fuzzy evolutionary algorithms and automatic robot trajectory generation", *Proc. of the First IEEE Conference on Evolutionary Computation*, 595-600.

[174] Zadeh L.A. (1992), "Foreword" of the *Proc. of the Second International Conference on Fuzzy Logic and Neural Networks*, Iizuka, Japan, XIII - XIV.

[175] Zadeh L.A. (1965), "Fuzzy Sets", *Information and Control*, 8(3), 338-353.

[176] Zhang C.D. and Song S.M. (1990), "Stability analysis of wave-crab gaits of a quadruped", *Journal of Robotics Systems*, 7(2), 243-276.

[177] Zhu Q. (1991), "Hidden Markov model for dynamic obstacle avoidance of mobile robot navigation", *IEEE Trans. on Robotics and Automation*, 7(3), 390-397.